The Dissertation Committee for Ashutosh Chakraborty
certifies that this is the approved version of the following dissertation:

Mechanical Stress and Circuit Aging Aware VLSI CAD

Committee:

David Z. Pan, Supervisor

Jacob Abraham

Michael Orshansky

Prashant Saxena

Nur Touba

Mechanical Stress and Circuit Aging Aware VLSI CAD

by

Ashutosh Chakraborty, B.Tech., M.S.E.E.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

December 2010

Dedicated to my father (Late) Shri Mrinal Kanti Chakraborty

and my mother Mrs. Meenakshi Chakraborty.

Acknowledgments

First and foremost, I would like to thank my PhD advisor Prof. David Z. Pan. Without his expert guidance this dissertation would not have been completed. He was readily available for technical discussions, and most of the skills that I have gained during my graduate student life, such as time management, research direction, problem solving, etc., I have learnt from him. Not only has he guided me through the most difficult research problems, but he has also provided psychological support during times of turmoil in my personal life.

I would like to acknowledge the help of my fellow graduate students. I am thankful to the technical discussions and friendly support they have provided. Past and current graduate students without whose help this dissertation would not have been possible include: Suhail Ahmed, James Ban, Minsik Cho, Jae-Yong Chung, Duo Ding, Jerrica Gao, Ou He, Wooyoung Jang, Anurag Kumar, Katrina Lu, Tao Luo, Joydeep Mitra, Jiwoo Pak, Anand Rajaram, Anand Ramalingam, Peng Yu, Jae-Seok Yang, Kun Yuan, Rein Zhang and Yilin Zhang. It has been my privilege to work with you all. Special thanks to Kun Yuan who provided me immense support throughout these years, both in times of deep despair and in times of joy.

In 2004, I came across a paper written by Prof. Marek-Sadowska from

University of California Santa Barabara on pre-placement wirelength estimation. The paper discussed physical concepts like contraction and attraction to derive high fidelity wirelength estimates. I was truly amazed by how well I could relate to the "physical" nature of the problem - like real objects trying to push/pull its neighbors. At that time, I decided to research algorithms targeting physical design. Since then, my academic life has always been intellectually exciting. Thank you, Prof. Marek-Sadowska.

I would like to express deep gratitude to my PhD committee members: Prof. Jacob Abraham, Prof. Michael Orshansky, Dr. Prashant Saxena and Prof. Nur Touba for agreeing to serve on my PhD committee despite their very busy schedules, and for their insightful comments and discussions.

Special thanks to the friendly and professional administrative staff at the ECE Department and CERC at the University of Texas at Austin (UT). In particular, I am grateful to the support and encouragement from Melanie Gulick, the ECE graduate program coordinator. During my first week at UT Austin, back in the Fall of 2006, an outgoing PhD graduate mentioned to me that Melanie was the biggest friend and well-wisher an international graduate student could have in the ECE Department. He was right.

I would like to specially thank Prof. Adnan Aziz, Prof. Sanjay Banerjee, Prof. Arjang Hassibi and Prof. Chris Mack under whom I took several graduate courses which helped me improve my research work. I would also like to thank Dr. David Newmark (AMD), Dr. Prashant Saxena (Synopsys) and Dr. Ruchir Puri (IBM TJ Watson) for giving me opportunities to be an intern

in their respective groups during the Summer of 2007, the Summer and Fall of 2009 and the Summer of 2010, respectively. These internships provided me invaluable exposure to semiconductor industry and research lab environment.

Finally, and most importantly, I am indebted to the love and support that I have received from my family members. In special, I am thankful to my parents, sister and Rennie, who made several sacrifices to ensure that I could pursue my dream of doing a PhD. When God snatched away my father in the last year of graduate school, I had decided to give up and return home. If not for the immense emotional support and constant encouragement that I got from my mother, I would not have been able to complete my PhD.

Mechanical Stress and Circuit Aging Aware VLSI CAD

Publication No. _____

Ashutosh Chakraborty, Ph.D.
The University of Texas at Austin, 2010

Supervisor: David Z. Pan

With the gradual advance of the state-of-the-art VLSI manufacturing technology into the sub-45nm regime, engineering a reliable, high performance VLSI chip with economically attractive yield in accordance with Moore's law of scaling and integration has become extremely difficult. Some of the most serious challenges that make this task difficult are: a) the delay of a transistor is strongly dependent on process induced mechanical stress around it, b) the reliability of devices is affected by several aging mechanisms like Negative Bias Temperature Instability (NBTI), hot carrier injection (HCI), etc and c) the delay and reliability of any device are strongly related to lithographically drawn geometry of various features on wafer. These three challenges are the main focus of this dissertation.

High performance fabrication processes routinely use embedded silicon-germanium (eSiGe) technology that imparts compressive mechanical stress to PMOS devices. In this work, cell level timing models considering flexibility to

modulate active area to change mechanical stress, were proposed and exploited to perform timing optimization during circuit placement phase. Analysis of key physical synthesis optimization steps such as gate sizing and repeater insertion was done to understand and exploit mechanical stress to significantly improve delay of interconnect and device dominated circuits.

Regarding circuit reliability, the proposed work is focused on reducing the clock skew degradation due to NBTI effect specially due to the use of clock gating technique for achieving low power operation. In addition, we also target the detrimental impact of burn-in testing on NBTI. The problem is identified and a runtime technique to reduce clock skew increase was proposed. For designs with predictable clock gating activities, a zero overhead design time technique was proposed to reduce clock skew increase over time. The concept of using minimum degradation input vector during static burn-in testing is proposed to reduce the impact of burn-in testing on parametric yield.

Delay and reliability strongly depend on dimension of various features on the wafer such as gate oxide thickness, channel length and contact position. Increased variability of these dimensions can severely restrict ability to analyze or optimize a design considering mechanical stress and circuit reliability. One key technique to control physical variability is to move towards regular fabrics. However, to make implementation on regular fabrics attractive, high quality physical design tools need to be developed. This dissertation proposes a new circuit placement algorithm to place a design on a structured ASIC platform with strict site and clock constraints and excellent overall wirelength. An

algorithm for reducing the clock and leakage power dissipation of a structured ASIC by reducing spine usage is then proposed to allow lower power dissipation of designs implemented using structured ASICs.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Challenges for Nanometer IC Design

For the last half century, technology scaling has been the primary workhorse leading to stellar advancement of the overall semiconductor industry. With each generation of technology scaling, most of the figure of merits of a VLSI product (such as performance, functionality, power dissipation, etc.) continuously improved. However, ever since the process technology has moved into sub 100-nm node, several very challenging issues have cropped up. State of the art fabrication technology currently has gate length in the range of a few tens of nanometer [15]. Scaling down to such small geometries brings forth a plethora of problems which make the process of device scaling exorbitantly expensive. Due to lithography challenges, physical scaling of dimension to be imaged needs extremely expensive equipments as well as exotic resolution enhancement techniques (RET). Due to short channel effect, the device performance does not improve as fast as it traditionally has. To obtain reasonable performance, threshold voltage scaling has not kept pace, leading to devices that consume nearly as much power while being idle as they do while performing useful computations. Supply voltage scaling has also slowed down leading to very high electric fields in the channel region leading to low reliability of the

devices. The device junction temperatures have increased as the technology nodes has advanced leading to need for expensive cooling equipment and negatively impacting device reliability. Overall, the process of realization of a high performance VLSI product which can reliably operate for several years and can be manufactured with reasonable cost, has become extremely challenging. These three objective i.e. performance, reliability, cost would be considered individually in the rest of this section.

One of the key alternatives to brute-force technology scaling for higher performance is to exploit process induced mechanical stress. The mobility of the charge carriers in the PMOS and NMOS devices can be enhanced by imparting suitable type (compressive vs tensile) of stress in the suitable direction (transverse, longitudinal) of the flow of charge carriers. Application of mechanical stress causes splitting and warpage of the six-fold degenerate conduction band of the charge carriers [73] in PMOS devices leading to larger drive current and hence lesser delay. For example, recently, [100] has reported the fabrication of PMOS devices with 200% improved mobility using multiple process induced stressing mechanisms. This phenomenal performance improvement is equivalent to that provided by several generations of technology scaling. Exploiting this mechanical stress for higher performance in devices is indeed an exciting opportunity to extract as much performance from a technology node without requiring physical dimension scaling.

Even if a high performance product is fabricated, it is difficult to guarantee its performance due to the so called "aging" effects. Such effects include

negative bias temperature instability (NBTI), positive bias temperature instability (PBTI), hot carrier injection (HCI), etc whose impact has been increasing with technology scaling owing to thinner gate oxide, reduced gate lengths and increasing electric fields in channel region. Among these, of particular interest is NBTI which is the primary device failure mechanism for PMOS devices. NBTI causes gradual increase in the threshold voltage of PMOS devices when they are negatively biased thereby reducing their drive current and hence performance. Existing works have shown that the NBTI effect can degrade the threshold voltage of the PMOS devices in the circuit by more than 25% over the lifetime of the product. For synchronous designs, the *clock* signal is the most crucial signal in the whole chip from timing as well as power considerations. Very precise clock distribution schemes aided by clock buffers are used for distributing high quality clock signal to obtain fastest possible circuit performance. However, power reduction schemes such as clock gating cause imbalance in the signal probability of the nets comprising the clock tree. This is because due to different clock gating duration of different parts of the clock tree, the duration of time negative bias is applied to different PMOS devices differ and so does their NBTI degradation and threshold voltage change. The difference in the threshold voltage of the clock buffers of the clock tree leads to different delay through them. This can lead to clock skew violation resulting in functional failures. Since these aging phenomenons increase tremendously at elevated temperature and voltage of operation, the process of burn-in testing is of serious concern. During the burn-in testing, a product is operated

briefly at voltage and temperature far greater than normal operating conditions. Therefore, aging degradation during burn-in testing is a big concern.

The finest layout features printed on a wafer these days are as small as 1/8th of the wavelength of light used for lithography. This "impossible" feat has been achieved by tremendous innovations by lithographers and fabrication facilities such as immersion lithography, heavy resolution enhancement techniques, optical proximity correction, double pattern lithography etc. Though these innovations have made it possible to print fine features, the cost of doing so has increased exorbitantly. Due to physical limits the amount of variability in the device parameter has also increased substantially as a fraction of their nominal values. Delay and reliability both strongly depend on geometry of various features on the wafer such as gate oxide thickness, channel length, contact position, etc. Due to increased variability in manufacturing process, this dependence creates severe problems in robust analysis and optimization considering mechanical stress and reliability effects. One of the key approaches to tackle this variability induced problems is to perform physical implementation on regular fabrics. Due to strict physical regularity of such fabrics, dimension variability can either be optimized away or pre-characterized leading to good correlation between analysis tools and on-silicon measurements. However, to make implementation on regular fabrics attractive, high quality physical design tools need to be developed. This dissertation proposes a new circuit placement algorithm to place a design on a structured ASIC platform with strict site and clock constraints and excellent overall wirelength. An al-

gorithm for reducing the clock and leakage power dissipation of a structured ASIC by reducing spine usage is then proposed to allow lower power dissipation of designs implemented using structured ASICs.

1.2 Overview and Contribution of This Dissertation

The algorithms and design techniques presented in this dissertation target the above mentioned challenges and are described in the next three chapters. The overall flow of the dissertation is as follows:

Chapter 2 discusses the techniques to exploit mechanical stress effects for higher performance at synthesis and layout level. The dependence of stress in channel of a PMOS device on the active area dimension for eSiGe technology is used to derive cell level timing models. Using these, we propose late ECO stage layout optimization by identifying and surgically replacing timing critical cells by higher mobility variants. At a higher level of abstraction, we also study the impact of mechanical stress on various physical synthesis stage steps such as gate sizing and repeater insertion and propose mathematical formulations to perform these steps concurrently while modulating the channel stress by active area dimension sizing.

To harden an IC design against the NBTI aging effect, we focus on clock tree design to withstand aging induced skew violations during its expected lifetime. We have proposed a runtime (circuit design) and design time (CAD) techniques to minimize the skew degradation by implementing clock gating using a mix of active high and active low clock gating elements. We have

also studied the detrimental impact that burn-in testing has on circuit delay degradation due to the process of burn-in testing. An algorithm for choosing a minimum degrading input vector during static burn-in testing is proposed to deal with this issue. These technique for controlling NBTI effect induced circuit aging are detailed in Chapter 3.

Several companies have come up with regular structures called "structured ASICs" which lie in between the ASIC methodology (characterized by no regularity, fast performance, low power dissipation, cheap in high volume) and FPGA methodology (characterized by complete regularity, slow performance, higher power dissipation, cheap to prototype). Moving to structured ASICs as the manufacturing fabric can considerably reduce printability, signal integrity, power/clock routing issues while providing reasonable performance and economic mid-volume product designs. To successfully migrate a design on to a structured ASIC, physical design tools considering the strict site, clock, and placement constraints of structured ASICs must be developed to minimize performance gap between them and ASICs. We have developed a placement engine for minimum wirelength placement of design to a structured ASIC considering all legality issues and minimum wirelength. To reduce the power dissipation of the resultant physical implementation of a product on a structured ASIC, we have also proposed low power placement. The key idea to reduce the power dissipation is to place the cells so as to maximize the part of the implementation platform that can be disconnected from power supply. The algorithms and implementation of both these placers and results are detailed

in Chapter 4.

Finally, in Chapter 5, we draw conclusions based on the results of the previous chapters as well as present a few promising future research directions to further investigate mechanical stress and reliability aware VLSI CAD algorithms.

Chapter 2

CAD Exploiting eSiGe Induced Mechanical Stress

2.1 Introduction

Last four decades have witnessed a tremendous increase in the performance of an integrated circuit (IC) while the cost of fabrication of each of these units has gone down considerably. The most important enabler of the success of the semiconductor industry is the continuous shrinking of the transistor device size which delivers faster, cheaper and smaller transistors in each generation. State of the art technologies currently have gate lengths as small as a few tens of a nanometer[15]. Scaling down to such small geometries brings forth a plethora of problems which make the process of device scaling exorbitantly expensive. These problems include lithography challenges, multi-million dollar mask costs, low yield ramp-up, and exponentially growing leakage current etc. In fact, since the introduction of the sub-100nm technology node, further device scaling has become extremely costly and technologically challenging. The need of improved performance and the challenges of physical scaling has mandated exploration of alternate techniques which improve performance of a transistor without requiring physical scaling. These techniques include the use of III-V elements (such as Ga, As), application of advanced

stress engineering techniques such as stress liners, use of carbon nano-tubes (CNT), 3-D IC integration, multi-core systems etc. Currently, each of these techniques is a very popular field of ongoing research. Among these, advanced stress engineering has emerged as one of the most promising techniques of device performance increase because unlike techniques such as use of CNT and III-V elements, the use of stress engineering can be seamlessly integrated in existing chip manufacturing process without the need of new materials or drastic manufacturing process modifications. In fact, stress engineering has already been used by several companies such as IBM, Intel and AMD to boost the performance of their VLSI chip. At the same time, stress engineering enhances *device* performance as compared to techniques such as 3-D IC integration which primarily increases the performance of a chip by reducing the *interconnect* length between communicating devices by placing them closer in the vertical dimension.

The application of mechanical stress alters the degeneracy of the energy bands in the channel of PMOS or NMOS devices which significantly changes the charge carrier mobility [19]. Mechanical stress can be of two types: compressive or tensile. In general, the mobility of a PMOS device increases when its channel is subject to compressive stress and decreases under tensile stress. For NMOS devices, an opposite trend is observed - tensile stress improves the device mobility whereas compressive stress degrades it. The objective of stress engineering is to intelligently apply the right kind of (i.e. compressive or tensile) mechanical stress to the channel of PMOS/NMOS devices in or-

der to obtain higher performance. There are several mechanisms of imparting mechanical stress to a PMOS or NMOS device. The primary stressing mechanisms of PMOS devices are source/drain silicon germanium (S/D SiGe), use of shallow trench isolation (STI), and stress liners. The primary stressing mechanisms for NMOS devices are STI and stress memorization technique (SMT). A brief description of each of these stressing mechanisms is as follows: S/D SiGe techniques etches out the silicon from the source and drain (S/D) regions of conventional transistor and epitaxially fills them with $Si_{1-x}Ge_x$ alloy where x denotes the proportion of Germanium in the alloy. Larger lattice constant of the $Si_{1-x}Ge_x$ alloy compared to that of Silicon creates compressive stress in the channel region. STI stress arises due to the ubiquitous method of using shallow trench of silicon oxide to isolate neighboring NMOS and PMOS devices. The stress generated due to STI is compressive in nature. Stress liners are highly stressed (compressed or tensile) silicon nitride liners which are deposited over the entire wafer to increase the carrier mobility [75]. Stress memorization technique (SMT) relies on depositing a stressed film on the wafer followed by rapid thermal anneal (RTA). Further details and a comprehensive review of these process-induced mechanical stress techniques can be found in excellent references such as [101][74].

In this work, we focus on the S/D SiGe technique which imparts compressive stress to the channel. The magnitude of the stress in the channel due to the S/D SiGe technique is critically dependent on the dimension of the active area around the device. We perform timing optimization of a design at the

layout level by exploiting this active area dependent mobility of S/D SiGe type PMOS devices. In particular, we propose a new active area stretching aware cell delay model and demonstrate its linear nature for the range of active area dimention for current and future generation devices. Based on this model, our technique transforms the timing optimization problem into a Linear Program (LP) formulation which can be solved for minimum cycle time or for timing closure under a given timing budget. For removal of the overlaps resulting from active area stretching, a fast minimally intrusive linear time legalization algorithm is presented. Our methodology is particularly attractive for late-mode ECO optimization since it directly works on the layout of the design. To enable dealing with hundreds of thousands of standard cells, this work is performed at the standard cell level of abstraction and the internal structure of the cell (such as necessary shift in contact position, IO pin reshaping) are abstracted out. We assume that it is possible to size the active area of a cell in a continuous fashion. In practice, for standard cell based designs, the cell library can have several variants of a cell with different active area dimension. In such a case, the active area sizing results obtained from our technique can be discretized to match available cell sizes.

Consider the side-view of two (series connected) SiGe S/D PMOS devices in Figure 2.1 shown without spacer (needed for lightly doped drain region to suppress short channel effects) for the sake of clarity. Note that the only difference as compared to a traditional PMOS device is that the S/D regions are filled with SiGe alloy which imparts compressive stress in the channel under

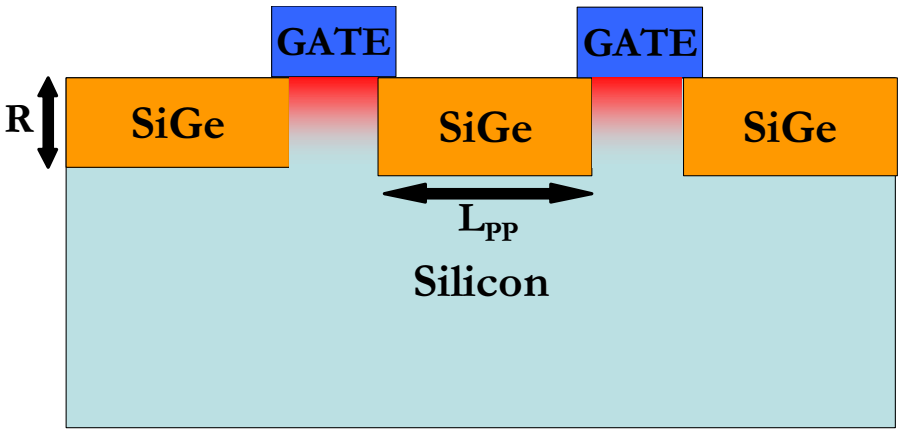

Figure 2.1: Side view of a SiGe S/D device. Source/Drain regions are epitaxially filled with SiGe which compresses the channel.

the gate polysilicon (poly). The length of active area adjacent to the device channel is equal to the poly to poly distance (L_{pp}) between adjacent poly gates, therefore we will use the term L_{pp} to denote the dimension of the active area in the rest of this section. The work in [42][24] has demonstrated that the stress in the device channel can be modulated by changing the active area dimension. In particular, [42] presented the transconductance improvement for isolated PMOS transistors as well as simulation results for densely packed transistors. Their results show significant improvement in transconductance and channel stress as a function of L_{pp}.

The hole mobility of PMOS devices has a near linear dependence on the uniaxial stress in the channel [87][76]. Using this, we transformed the increase in the stress values to the increase in mobility. SPICE simulations were then performed with the modified mobility values to obtain the delay

of an inverter as a function of the channel stress. The nominal value of L_{pp} for the 45nm technology transistor was taken as 100nm. Figure 2.2, shows the improvement in the delay of the PMOS device as compared to a device without any increase in L_{pp}. For these set of experiments, [42] used 25% Ge in SiGe alloy (i.e. $Si_{0.75}Ge_{0.25}$), a recess depth (dimension R in Figure 2.1) of 120nm, gate length of 45nm, and nominal L_{pp} of 100nm. We refer readers to the cited work for process related details of the experiments carried out.

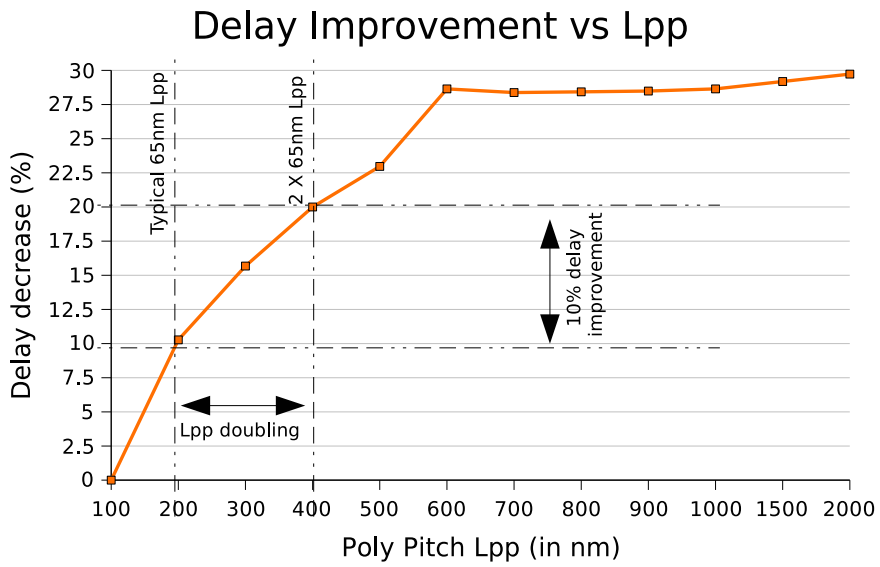

Figure 2.2: Delay improvement of PMOS device as a function of active area dimension (L_{pp}). Delay improves by 10% when L_{pp} is doubled from its nominal (for 45nm DRC) value of 100nm to 200nm.

The near-linear dependence of the performance improvement on L_{pp} is evident from Figure 2.2 which saturates at L_{pp} exceeding 600nm. The design rules for 65nm, 45nm, and 32nm gate lengths require the contacted L_{pp} to be

approximately 150nm, 100nm, and 70nm respectively [15]. Even if we allow tripling of these L_{pp} dimensions for performance improvement, it is still below the saturation limit of 600nm. We performed a linear fit (with R^2=0.96) of the performance improvement vs L_{pp} increase curve in the range of interest of 100nm to 200nm and 200nm to 400nm and we observed that the doubling the active area size roughly corresponds to 10% better PMOS performance. We would like to point out that using S/D SiGe only increases the mobility of PMOS devices which improves the rise time of the cell without impacting its fall time. Since the delay of a cell is the average of fall and rise time, whenever we report reduction in the delay of a cell, we do it after scaling by a factor of 0.5 to compensate for improvement of only PMOS devices. This method is reasonable for most cases since in a timing path, nearly half of the cells are undergoing 0 => 1 transition while the rest are switching the other way. For a rare design whose critical paths have mostly all rising transition or all falling transition cells, the timing improvement should be obtained by performing static timing analysis.

2.2 Late Stage Layout Optimization Considering Stress

In this section, we will present a late stage (i.e. post routing) layout optimization methodology to best exploit the active area dependent mobility of eSiGe devices. The basic idea that we exploit is to trade-off the white space in the vicinity of timing critical cells by replacing the timing critical cells with their variants which have a larger active area size. The challenge for such

an approach is to come up with a high fidelity mathematical formulation for determining which cell should increase its active area by how much, and how to enhance the predictability of the timing improvement by reducing the chances that the process of placement legalization might generate new critical paths.

Consider the top view of a transistor before and after L_{pp} resizing in Figure 2.3 such that the L_{pp} after resizing is twice its original value. Each of the vertical tall (red) bars are poly gates, and the horizontal (yellow) tile is the active area. On increasing the L_{pp}, the delay through the gates can be reduced. Previously we saw that doubling the L_{pp} of a transistor can decrease its delay by 10%. Since in our methodology we stretch standard cells, in this section we will derive the effective active area increase as a function of increase in standard cell width (instead of increase in L_{pp} distance). It is visually evident that the increase in standard cell width is not 2X, but less than 2X when the L_{pp} is increased to 2X of its original size. Conversely, doubling the width of a standard cell effectively increases the L_{pp} dimension by more than 2X.

Let us assume that the gate length is L_g and the oxide isolation thickness on each side of the active area is L_i. Note that the gate length or the insulation oxide thickness does not increase while sizing up the active area. The total width of the standard cell before stretching is W. We considered the case where there are n poly gates (i.e. n=2 for 2-input NAND/NOR, n=3 for 3-input NAND/NOR etc.) in the cell. The original width of the standard cell W in terms of constituent dimensions can be computed as

$$W = (n+1)L_{pp} + nL_g + 2L_i \tag{2.1}$$

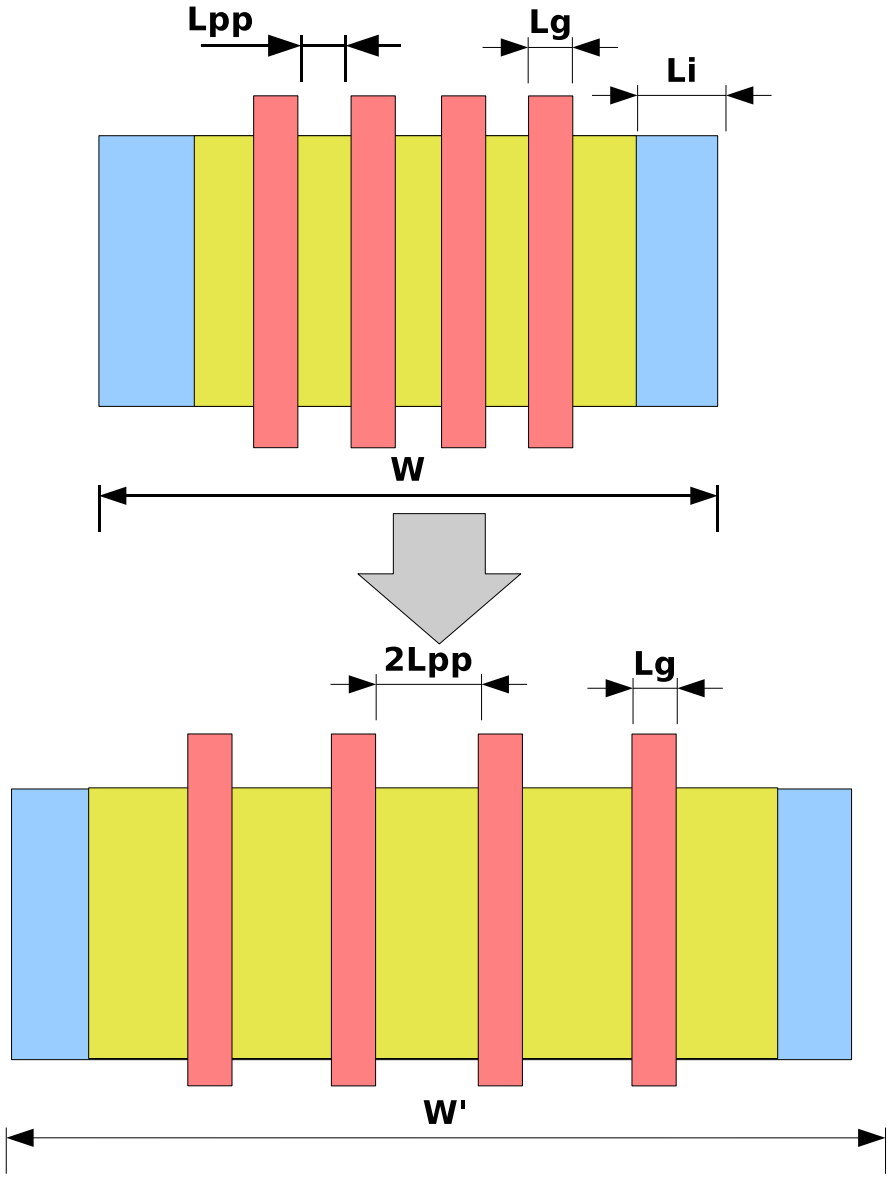

Figure 2.3: A scenario of doubling of L_{pp} of a cell and its effect on cell width. Notations: gate length (L_g), active area (L_{pp}), insulation oxide thickness (L_i), W and W$'$ are widths before and after active area doubling.

After increasing the width of the cell K times, if the L_{pp} increases to $L_{pp}^{'}$, then the following equation holds:

$$KW = (n+1)L_{pp}^{'} + nL_g + 2L_i \tag{2.2}$$

Taking the ratio of the above, we can derive that

$$L_{pp}^{'} = KL_{pp} + \frac{(K-1)(nL_g + 2L_i)}{(n+1)} \tag{2.3}$$

The above equation shows that increasing the width of a cell by K times, L_{pp} increases by more than K times. For example, in a typical case where insulation (i.e. L_i) and total gate length (i.e. $n \times L_g$) are 20% and 30% of W respectively, we can achieve 3X increase in L_{pp} by doubling the width of the standard cell. For each standard cell in our library, we stored the mapping between the increase in the width of a standard cell and the corresponding increase in its L_{pp} dimension which in turn was used to compute the delay decrease. Using Eqn 2.1, Eqn 2.2 and Eqn 2.3, if the delay of the cell when its width is increased to K times is D(KW), then it can be represented in terms of its nominal delay D(W) at original width of W as

$$D(KW) = D(W)\left(\frac{1 - \alpha(K-1)W}{(n+1)L_{pp}}\right) \tag{2.4}$$

where, the parameter α is the decrease in delay by doubling the L_{pp} which has been found to be 10% (i.e. 0.1). Note that the above expression only depends on the layout, the original width and the original L_{pp} of the device and thus is readily known once the cell library analysis is complete. In the

above discussion, we have assumed that all PMOS devices inside a cell benefit equally due to the active area sizing. In practice, the devices on the boundary of the cell improve more than the device in the center of the cell because the nominal value of stress near the center of the cell is already higher than that at the cell boundary. For higher accuracy, the above analysis can be done for each input pin of a cell rather than for the whole cell.

In the next three subsections, we present our layout level active area stretching based timing closure scheme. First we discuss the constraints on L_{pp} stretching which we enforce to bring *predictability* to the achieved timing improvement. The formulation of linear program (LP) is tackled next. Finally, we discuss the legalization algorithm to efficiently remove overlaps caused due to L_{pp} stretching.

2.2.1 Stretching Constraints for Predictability

Increasing the width of a critical cell[1] introduces overlap with non-critical cell adjacent to it and needs to be legalized. If this overlap is not properly controlled, the legalization step would cause substantial change in the location of non-critical cells. This increases the chances of turning a non-critical path into a critical path and can lead to several iterations of timing improvement without any guarantee of convergence. To control this effect, we propose the following constraints which our optimization flow must adhere to,

[1]We define a cell lying on a failing path as a *critical* cell here onwards. A cell which is not critical is referred to as non-critical cell.

while deciding about the extent of the streching of a critical cell. Due to the stretching of critical cells,

1. No critical cell can move (thus, can only stretch),
2. No non-critical cell can jump over a critical cell,
3. No cell (critical or not) should leave its row, and
4. A critical cell can stretch only until a particular limit.

The above rules are graphically represented in terms of valid and invalid movement in Figure 2.4. The original layout for the first three cases is given in Figure 2.4-A.

The first constraint enforces that the interconnect delay between any pair of critical cells remains unchanged after expansion. In Figure 2.4-B, the coordinates of cell A and C have changed, but that of critical cells B and D have not (even though these cells have stretched). Therefore, if there is an interconnect between cells B and D, the delay on that interconnect would not change.

The second constraint prevents "wide" movement of a non-critical cell which circumvents the potential problem of large increase in interconnect delay of nets incident on non-critical cell. The intra-row movement in Figure 2.4-C represents this invalid move since the non-critical cell C needs to jump over critical cell D to allow D to expand.

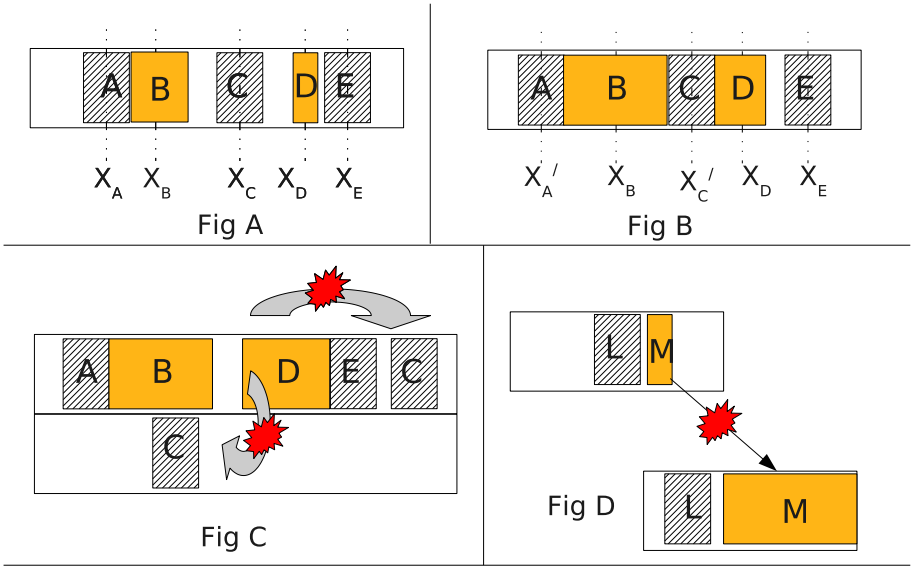

Figure 2.4: Graphical representation of the rules which determine the stretching of critical cells. Long horizontal slabs are circuit rows. Cells in bold orange color are critical cells. Non-critical cells are in hatched slanted line. Position X_i represents the center of mass of cell i. See adjoining text below for a description of the four figures.

The third constraint is like the second constraint with an added advantage that the modifications done in a row cannot induce detrimental changes to neighboring rows. The inter-row movement in Figure 2.4-C shows an example of invalid move since the non-critical cell C need to moved into another row. This constraint also removes any chain effect of overlaps in one row causing further overlaps in other rows.

The fourth constraint is a trade-off between the extent of layout modifications and the cycle time improvement in light of the saturation of improvement for large L_{pp} values (see Section 2.1). The stretching depicted in

Figure 2.4-D is invalid if the cell M is allowed to stretch up to 2X original size. In our experiments we enforce that no cell can be stretched to more than twice its original size.

2.2.2 LP Formulation for Timing Optimization

Consider a timing failing path p in the circuit. This path (as any other path) consists of an alternating sequence of standard cells and interconnects. Since the path is failing, all the cells will be critical (and thus, under purview of being stretched). Let the set $C_i = c_i^0, c_i^1 \; .. \; c_i^m$ be the cells in the path i. Since the critical cells never move from their original location (as per Rule 1 above), the interconnect delays between these cells can be summed up and taken as fixed before and after stretching. Let d_i^I represent the total interconnect delay for path i. Further let the delay of cell c be D_c (pin-to-pin delay). Thus, the total delay of the path i ($= DELAY_i$) can be written as:

$$DELAY_i = d_i^I + \sum_{j \in C_i} D_j \tag{2.5}$$

Recall that the delay of cell c can be written according to Equation 2.4. Let the width of the standard cell c with n_c poly gates be increased by ΔW_c. The path delay consisting of such cells can thus be expressed as

$$DELAY_i = d_i^I + \sum_{j \in C_i} \left(D_j(W) \times \left(1 - \frac{0.1 \times \Delta W}{(n_c + 1) L_{pp}^c} \right) \right) \tag{2.6}$$

where $D_j(W)$ is the original (i.e. without any increase in L_{pp}) of cell j. Note that the expression $(n_c + 1) \times L_{pp}^c$ is constant for a given type of cell and can be pre-computed and stored in a look-up table beforehand.

Achieving Highest Performance: If the optimization target is to achieve the highest possible performance while satisfying all the active area stretching constraints of Section 2.2.1, a linear program (LP) can be formulated as follows: Let P_{crit} denote all failing paths, the whitespace available between two consecutive critical cells a and b in a row excluding the space used up by non-critical cells between them be denoted by WS_{ab} and each such consecutive pair of critical cells be part of the set $PAIRS$. The set of all critical cells is denoted as $CRIT$. The LP is:

$$\begin{array}{lrll} \max & M & & \\ \text{s.t.} & DELAY_i + M & \leq 0 & (i \in P_{crit}) \\ & \Delta W_a + \Delta W_b & \leq WS_{ab} & (ab \in PAIRS) \\ & \Delta W_x & \leq W_x & (x \in CRIT). \end{array}$$

The dummy variable, M, when maximized such that the first set of constraints are satisfied minimizes the delay of the longest path in the circuit. The second set of equations force a solution in which there is always enough space between two critical cells so that cells lying between them can be locally moved without overlap, thus never violating Rules 2 and 3 of Section 2.2.1. Third set of equation is to prevent any cell to become more than twice its original size: in adherence to Rule 4. The objective function, when maximized, is equivalent to choosing the right values of amount of expansion for each cell (basically ΔW_x for each cell x that is critical) so that the circuit is the fastest possible.

Achieving Target Performance: If the optimization target is to meet a given timing requirement with least possible active area increase, the

LP can be formulated as follows: let the target path delay to be achieved be D_{target}, which is set by the required frequency of operation demand. The LP is:

$$\begin{array}{rrll} \min & \sum_{ab} (\Delta W_a + \Delta W_b) & & (ab \in PAIRS) \\ \text{s.t.} & DELAY_i & \leq D_{target} & (i \in P_{crit}) \\ & \Delta W_a + \Delta W_b & \leq WS_{ab} & (ab \in PAIRS) \\ & \Delta W_x & \leq W_x & (x \in CRIT). \end{array}$$

This formulation is different from that for achieving the highest performance because the cost function now tries to achieve the targeted performance with minimum increase in total active area. This helps to minimize the power penalty of active area sizing or to reduce the perturbation in the layout. In case a particular part of the layout is very sensitive and should not be disturbed, the ΔW's of the cells in that region can be set to zero.

Typically, a design requires some whitespace which subsequently is used by spare cells, decaps and filler cells. To accommodate these cells, an extra constraint can be added to the above two LP formulations to limit the maximum amount of active area growth to a particular fraction of the available whitespace.

LP Formulation Complexity Analysis: The total number of path delay constraints (first set) in both the above LP formulations is linear in the number of failing paths. The set $PAIRS$ only has pairs of *critical* cells which are in the *same row* of the layout such that there is no other critical cell between them (for example cells B and D in Figure 2.4). Therefore, the number of pairs in this set is linear in the number of critical cells. Similarly number of last set of constraints is also linear in the number of critical cells.

Therefore, the overall number of constraints in the LP formulation is O(C_{crit} + P_{fail}) where C_{crit} is the number of cells which are critical and P_{fail} is the number of paths that are violating timing requirement. The output of the LP is the amount of increase in the width of various critical cells.

2.2.3 Minimum Perturbation Legalization

After the LP has been solved and the critical cells enlarged, there may be overlaps between the expanded critical cells and their neighboring non-critical cells. This overlap needs to be removed through legalization. In view of the general philosophy of perturbing the minimum amount of interconnects and cells, we propose Algorithm 1 which remove overlaps while shifting the least number of cells by minimum displacement to get a legalized placement.

Algorithm 1 LegalizeDesign

```
for all critical cell C in the design do
    RemoveOverlapOn ( right ) edge of ( C )
    RemoveOverlapOn ( left ) edge of ( C )
```

The above algorithm makes two calls to RemoveOverlapOn (described in Algorithm 2) for each timing critical cell in the layout: the first time to remove overlap from the left edge of the critical cell and the second time to do it from the right edge. Next we describe the algorithm RemoveOverlapOn.

Consider the case in which Algorithm 2 is invoked to remove overlaps from the left edge of a critical cell. This means that the argument *DIR* is set

Algorithm 2 RemoveOverlapOn [direction DIR, Cell C]

1: Identify neighboring cell N on $DIR of $C
2: Let WS_{CN}$ be pre-stretching whitespace b/w $C and $N
3: Let ΔW_C be the increase in width of $C
4: Overlap $OL = ΔW_C/ 2 - WS_{CN}$
5: **while** $OL > 0 **do**
6: Shift $N by $OL to $DIR
7: $C $\Leftarrow$ $N
8: $N $\Leftarrow$ neighboring cell on $DIR of $N
9: Recompute WS_{CN}$ using pre-stretching location
10: $OL = $OL - WS_{CN}$

as *left*[2] The explanation of the complimentary case for the *right* direction is similar in nature. First, we identify the cell lying immediately on the left of the critical cell. In case the critical cell is the first cell of the circuit row, the beginning of the row acts as a dummy neighboring cell. Then, based on the increase in the width of the critical cell (Line 3) and the existing whitespace between the critical cell and the neighboring cell (Line 2), we compute the current value of overlap (Line 4). This value is the distance the neighboring cell must move in order to become non-overlapping with the critical cell. Next, we iterate until the value of overlap is non-negative. In each iteration, we shift the neighbor cell by the amount of overlap. We then assign the neighbor cell as the new critical cell whereas the cell next to the neighbor cell becomes the new neighbor cell (Line 8). Further, we update the value of overlap by subtracting the whitespace which originally existed between the new critical

[2]To better understand Algorithm 2, read it while replacing every occurrence of "DIR" by "left".

and the new neighbor cell. The loop terminates when the overlap is non-positive meaning that the extra width has been successfully consumed by the whitespace available. Figure 2.2.3 depicts the sequence of our legalization flow: the original layout and the layout after critical cell expansion are shown in a) and b) respectively. In Figure 2.2.3 c) and d) the overlap on right edge and left edge of each critical cells are resolved, respectively.

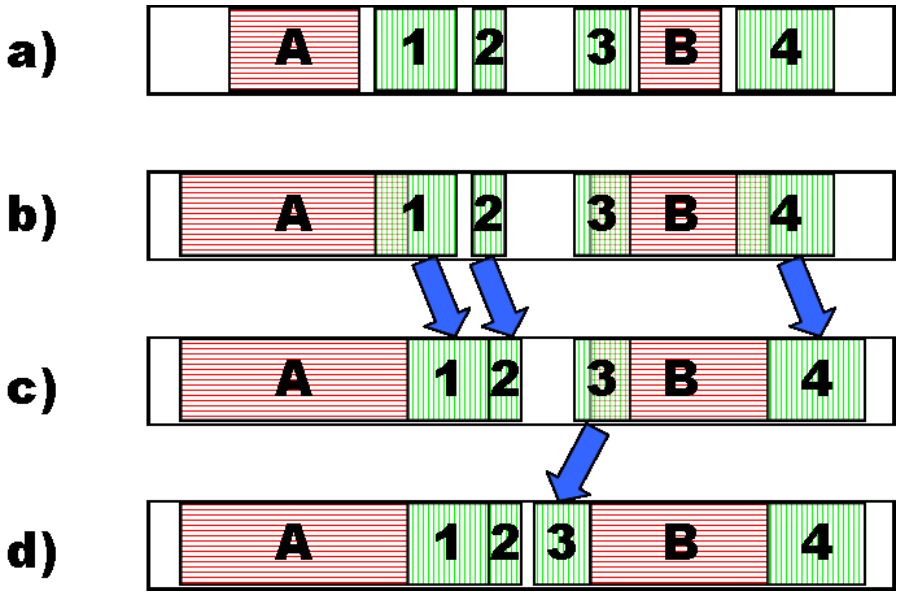

Figure 2.5: Sequence of four images depicting a simple standard cell row with two critical and four non-critical cells undergoing overlap removal. Critical cells are shown in red with horizontal lines. Non-critical cells are shown in green with vertical lines. a) shows the original layout, b) after critical cells have expanded, c) after removing overlap on right side of each critical cell, and d) after overlap removal on left side of each critical cell.

2.2.4 Experimental Flow and Setup

Figure 2.6 depicts our overall flow. Given a benchmark, we compile the RTL on to a standard cell library to technology dependent netlist using Synopsys Design Compiler. The next step is to perform place and route (PnR) of this design followed by parasitic extraction performed using Cadence SOC Encounter [26] (version 6.2) tool. At this stage, we generate an estimate of the active and leakage power dissipation of the design using the same tool (edge marked *Original Power* in the figure) based on the probabilistic switching activities at the circuit input. Parasitics aware timing analysis is then performed using Synopsys Primetime tool to identify the paths which need to be fixed as well the design cycle time (edge marked *Original Timing* in the figure). In our flow, we considered all paths whose delay is more than 80% of the maximum delay as failing.

A C++ program was developed to analyze the delay through the cells on the failing path along with their geometry and location in the chip layout to generate the set of constraints outlined in Section 2.2.1. The maximum allowed post-stretching width of each cell is constrained to twice of its original width. Based on these constraints, the two linear program (LP) formulation for minimum cycle time and minimum perturbation timing closure are generated as outlined in Section 2.2.2. These two LP are solved using the open-source *GNU Linear Programming Toolkit* [6] to compute the new width of cells on critical cells as well as the achieved cycle time of the design (edge marked *LP Timing* in the figure). Another C++ program was used to implement the active area

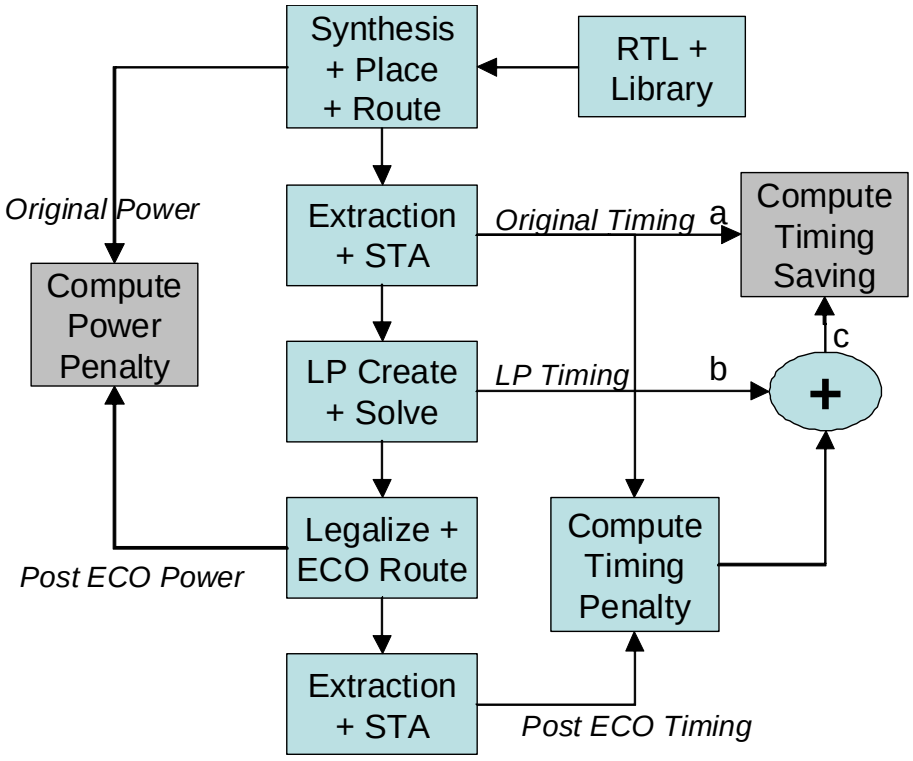

Figure 2.6: Flow used in our experiments.

sizing solution generated by the LP using the legalization procedure presented in Algorithm 1. In addition, this program un-routes all the interconnects which are incident on the cells that are moved due to the legalization process. ECO routing of the resultant design is then performed using Encounter tool to complete the routing of these nets followed by parasitic extraction of the changed layout.

To estimate the power dissipation of the circuit after active area sizing, we first use Cadence Encounter [26] tool to report individual nominal dynamic and leakage power of each gate. These power numbers are post-processed as

follows: The dynamic power of the cells which undergo active area sizing are scaled by the increase in their active area to account for larger diffusion capacitance that needs to be charged/discharged while switching. For leakage power estimate, we generated a look-up table which maps the increase in mobility of a PMOS device to the increase in its leakage power through SPICE simulations. The leakage power numbers of the cells which undergo active area sizing are then scaled by the factor corresponding to its mobility increase from the look-up table generated. The resulting power dissipation is depicted as the edge marked *Post ECO Power* in the figure which is compared with the original power dissipation to compute the increase in power dissipation.

After the ECO routing is complete, we perform a parasitics aware timing analysis using Primetime to report the new timing of the design (edge marked *Post ECO Timing* in the figure). The difference between the timing achieved at this step and that of the original layout represents the penalty paid due to perturbation of the original solution generated by placer/router. This timing penalty is added to the timing reported within the C++ timing engine (marked *LP Timing*) to compute the total effective timing of the design and thus the timing saving achieved by our method is computed.

Example 1. Imagine a circuit which has been placed and routed. Parasitic aware timing analysis yields the maximum delay through the circuit as 10ns. We run the critical paths (say which have a delay more than 9ns) through the LP formulation for active area stretching. The solution of LP reports the improved delay through the circuit as 9.5ns. We perform legalization and ECO

routing on the solution provided by the LP and rerun parasitic aware timing analysis which reports the delay as 10.1ns. Comparing 10ns and 10.1ns, we compute the timing penalty due to legalization and ECO routing as 0.1ns. This penalty is added to 9.5ns to get the corrected delay of optimized circuit as 9.6ns which when compared to original delay of 10ns boils down to 4% reduction in the delay of the circuit.

All experiments were performed on 64-bit 4-CPU 8-GB RAM machines running Linux operating system. Our code was implemented in the C++ language. We performed two sets of experiments to demonstrate the efficacy of our optimization technique. The first set of experiment is targeted to explore the dependency of our optimization method on the choice of benchmark circuits. The second set of experiment deals with the optimization of one particular benchmark for a variety of different design parameters. The method of estimating routing effort of a design and the setup of these two set of experiments are explained next.

2.2.5 Estimating Routing Effort

To faithfully capture the detrimental impact of our optimization flow on the timing due to increased interconnect length, it is important to consider the routing *effort* of the design. Routing *effort* can be defined as the difficulty a router faces to successfully route all the nets and is a function of average number of fanouts of cells and degree of connectivity among the cells. A design with high routing effort is very susceptible to movement of cells during the

legalization step since the resulting ECO routing may introduce significant detour to re-route it, possibly generating a high delay path. Traditionally, routing effort is measured by routing congestion which is computed on each global routing grid cell perimeter as the ratio of interconnects crossing the grid perimeter to the number of tracks available. However, this per-grid routing congestion is not suitable for our purpose since we want to explore the impact of higher overall routing effort of the design. To capture this, in our experiments we used the routing layers available in the router (during the initial routing phase and ECO routing phase) as a proxy for modulating the holistic routing effort of the design. The reduction in available routing layers can be considered as reducing track capacity per grid cell summed over all routable layers thus measuring the resources available to the router.

2.2.6 Experiment Set 1

In the first set of experiments, our aim is to quantify the optimization potential of our methodology on a variety of different benchmark circuits. Since these benchmarks have different characteristics such as cell/net count, path delay distribution, number of inputs/outputs, and interconnect structure, this experiment proves the generic nature of our optimization method. All the properties of the optimization flow were kept similar for all the benchmark circuits in order to avoid any bias. These properties include layout row utilization, available routing layers, aspect ratio of the die, and buffers/repeaters available to the physical synthesis tool. Table 2.1 shows the primary char-

acteristics of the different benchmarks we used for this set of experiments. The geometrical aspect ratio of all layouts was fixed to be 1.00 and the circuit row utilization was set to be 70%. The technology independent RTL of these benchmarks are obtained from the open-source *Opencores* [12] design repository. All designs were technology mapped to the open-source 45nm non-linear delay model (NLDM) library from *Nangate* [11] and were routed using 7 routing layers which is the maximum available layers.

Bench Ckt	**Num Cells**	**Num Nets**	**Num IO**	**Avg Fanout**	**Row Util**	**Route Lyrs**
des	88566	90660	298	2.10	71.1%	7
vga	4128	4761	214	2.26	70.4%	7
dlx	13471	14678	170	2.47	72.0%	7
ethernet	49427	50460	210	2.36	69.3%	7
i8051	9752	10432	100	2.63	70.7%	7
JpgComp	12294	13324	77	2.38	69.4%	7

Table 2.1: Characteristics of the benchmark circuits used in the first set of experiments. The columns denote the number of cells, nets, IO pins, average fanout of each cell, layout aspect ratio, row utilization, and number of layers allowed for routing. Aspect ratio of die for layout was set as 1.00.

2.2.7 Experiment Set 2

In the second set of experiments, we narrowed our focus to a single benchmark circuit and performed an in-depth analysis of how the timing results achieved differ due to various typical design choices. The parameters that were varied for this benchmark are: standard cell library (high or low V_{th} variants), routing effort, and core row utilization. The choice of threshold

voltage determines the path delay distribution. We used two variants of a commercial 90nm standard cell library for this set of experiments: high V_t (350mV) and low V_t (200mV). The nominal supply voltage of the library was 1V. The choice of core row utilization defines the amount of whitespace available for cell stretching. We performed circuit layout with row utilization values ranging from 20% to 85% in steps of 5%. The high routing effort version of the layout was allowed only 4 routing metal layers to complete all the routing whereas the low routing effort version had the freedom to route with 7 routing metal layers. Table 2.2 shows the primary characteristics of the benchmark we used for this set of experiments.

Bench Ckt	**Num IO**	**Num Cells**	**Num Nets**	**Avg FO**	**Num Cells**	**Num Nets**	**Avg FO**
		Low Vth			**High Vth**		
wims	112	12238	22492	3.67	11454	21209	3.70

Table 2.2: Characteristics of the benchmark circuit used in the second set of experiments. The columns denote the number of IO pins and the tuple {number of cells, nets and average fanout of each cell} for high and low V_{th} library mapped variants.

2.2.8 Results for Experiment Set 1

We report the results for the two formulations: achieving the highest performance where the objective is to make the circuit as fast as possible and for achieving target performance where the objective is to obtain a particular performance with minimum stretching (corresponding to minimum dynamic power increase).

Table 2.3 shows the results achieved by our flow for various benchmarks where the objective is to make the circuit as fast as possible. Columns a, b, and c tabulate the cycle time of original design, LP reported cycle time before legalization/ECO routing (legalization here onwards), and final timing considering the correction due to perturbation arising from legalization. In Figure 2.6, these correspond to edges marked a, b, and c respectively. The timing improvement achieved before legalization is in Column d and their corresponding values after legalization is in Column e. Column f shows the difference between these two timing improvement values. Columns g, h, i, j, and k show the number of critical cells, number of cells stretched, number of cells moved during legalization, increase in dynamic power consumption, and increase in leakage power consumption of the chip respectively. Column l shows the number of nets that need to be ECO routed.

From Table 2.3 we observe that across different benchmarks, our technique can improve the cycle time by as much as 6.77% and as little as 4.5%. Averaging across different benchmarks, we obtain 5.74% cycle time improvement which corresponds to more than 6% increase in the frequency of operation of the design, which is remarkable at post placement stage without requiring any change in the netlist structure.

We note that the difference between the delay improvement predicted by the solution of LP (i.e. before legalization) and the actual saving after legalization is very minute with an average value of 0.06%. In fact, in two of the six benchmark, there is no difference between the two timing improvements.

This is a key observation because the very small difference means that the designer can get immediate feedback of the potential timing improvement right after solving the LP without the need of legalization or ECO routing. In case the timing achieved is not sufficient, the maximum size to which a cell is allowed to stretch can be increased and our optimization flow can be re-run. On the contrary, if the timing achieved is significantly high, the criteria of failed path definition can be changed to capture additional paths and our flow can be run again. In this way, our flow is very suitable as an efficient *what-if* analysis tool. The very small difference due to legalization and ECO routing is enabled due to the rigid stretching constraints we propose in Section 2.2.1. Without these constraints, the LP reported timing improvement would be higher but there would be much more uncertainty regarding the final timing achieved due to larger perturbations during legalization and ECO routing. Figure 2.7 shows the savings achieved before and after legalization for different benchmarks (also shown in Table 2.3).

From Table 2.3, we also observe that very few of the critical cells undergo active area stretching. On an average, only 0.42% of n their width which increase the dynamic power of the design by 0.58%. The average increase in leakage power of the design is less than 5%. Notice that this is 10X more than the impact on dynamic power because mobility enhancement has a much stronger impact on leakage than dynamic power or delay. The number of cells that move during legalization because of stretching of critical cells is only 0.37%. Due to the movement of cells during legalization, on an average

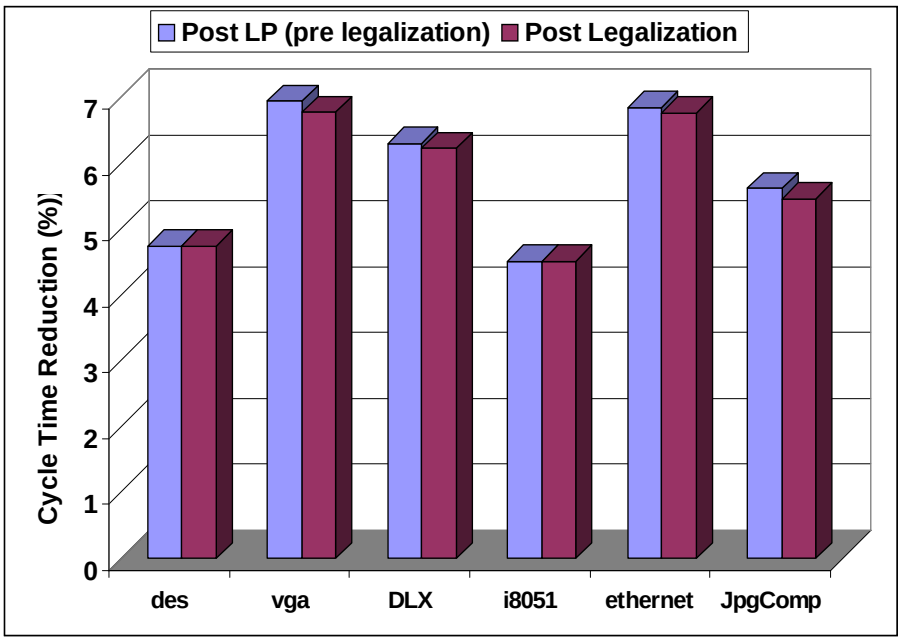

Figure 2.7: Cycle time reduction achieved by our flow just after LP solving and after (legalization+ECO routing). Up-to 6.5% cycle time reduction is observed. Minute difference between the saving numbers allows early prediction of the optimization potential.

2.4% of the nets need to be ECO routed. Except the benchmark `JpgComp`, the LP solving, legalization and ECO routing for all other benchmarks finished within 15 minutes. For `JpgComp`, the router spends a longer time due to a larger number of nets (5.46%) that need to be re-routed.

Next, we present the results for the formulation which achieves a given timing target with minimum increase in dynamic power. Note that the value given in Table 2.3 is the upper bound on the improvement of timing that can be obtained which corresponds to letting all the cells increase their width as long

as the timing is improved while satisfying stretching constraints. We incrementally tightened the targeted timing (T_{tgt}) of the design in steps varying from 1% to 5% and observed very minute difference between the timing reported before legalization (i.e. after LP solving) and after legalization (<0.04%). Therefore only the final delay numbers after legalization and ECO routing are reported. Table 2.4 tabulates the results obtained: the second and the third column show the original timing of the design and the number of critical cells in the failing paths respectively. Each set of three columns thereafter a, b, and c reports the number of cells stretched, the number of cells that move during legalization, and the increase in the dynamic power of the design, respectively, while achieving the corresponding T_{tgt}. The layout parameters such as aspect ratio, row utilization and the number of routing layers are the same as that for Table 2.3.

The first observation we make from Table 2.4 is that *very* few cells need to stretch and move during legalization as compared to the case when the objective is to achieve maximum performance (in Table 2.3). This is due to the explicit objective function which minimizes the stretching for achieving a target performance. For example, in benchmark `des`, only 4 cells need to stretch in order to achieve 96% target delay (meaning 4% performance improvement) whereas to achieve a performance improvement of 4.74%, 48 cells need to stretch, a 12X increase. Similarly, for benchmark `JpgComp`, the power penalty to achieve 5.46% delay improvement is 2X (0.70% from Table 2.3 vs 0.338% from Table 2.4) as compared to achieving 5% delay improvement. Therefore,

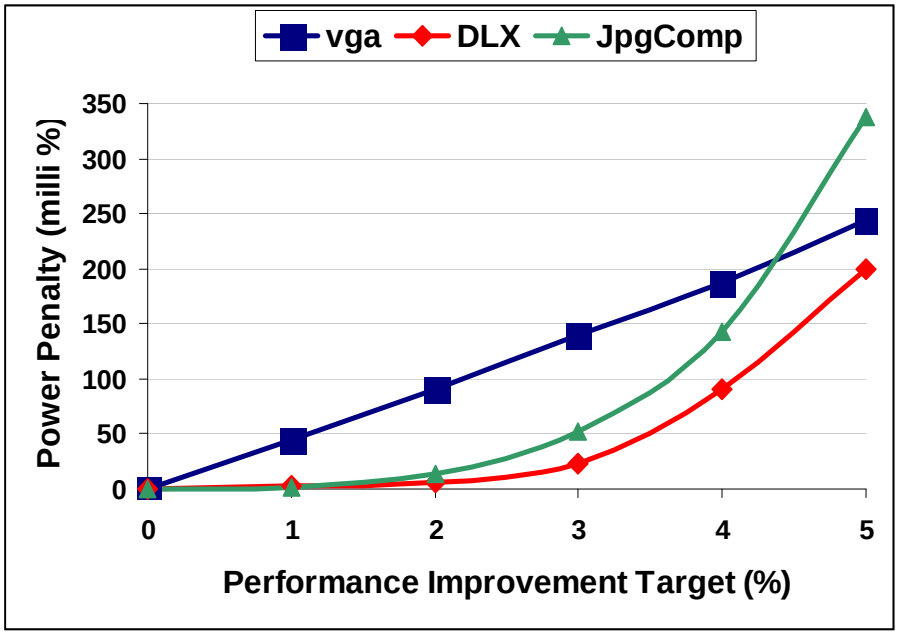

Figure 2.8: Dynamic power penalty (due to larger diffusion capacitance of increase active area) as a function of performance improvement for some of the benchmark circuits. Y-axis is in milli% = 0.001%. Values of other benchmarks can be read off Table 2.4.

we conclude that one should use the formulation for achieving maximum performance when absolutely necessary as the resultant layout modification can be much more for small incremental performance improvement. For benchmarks `des` and `i8051`, it was impossible to achieve 5% reduction in cycle time as the maximum saving for these benchmarks were 4.74% and 4.51% respectively. In Figure 2.8, the dynamic power penalty of some of the benchmarks are plotted as a function of the different target performance. There is a smooth trade-off between the target performance and the power penalty.

Continuous vs Discrete Active Area Sizing: For standard cell based designs, the choice of active area available during layout is restricted to the variants available in the library. The performance improvement shown in this section has so far assumed that the active area of a cell can be sized in a continuous fashion. To understand how the availability of only a few discrete active area sizes of each cell in the library well affect the optimization potential, we repeated our experiment after restricting the choice of cells available. This effectively changes our formulation from linear programming to integer linear programming. Figure 2.9 shows a radar plot of performance improvements achieved using discrete choice of active area sizes normalized w.r.t. the savings achievable by using continuous choice of active area sizes. Each radial line represents a benchmark. The data is shown for the cases when there are 2, 3, 4, or 5 variants of each standard cell available in the library. For the case with n variants, the available expanded active area sizes are assumed to be equally spaced between 1X (i.e . the nominal size) to 2X (i.e. twice the nominal size). For example, for the case when 3 variants are present, the active area size of these variants are 1.0X, 1.5X and 2.0X times the nominal size. From Figure 2.9, we observe that some benchmarks such as `Jpgcomp` and `vga` are not very sensitive to how many variants of each cell exists in the library and even if there are only 2 active area sizes available (1X and 2X), the savings achieved is almost as good as that achievable with continuous sizing. On the other hand, the timing improvements achievable for benchmarks `DLX` and `ethernet` are very sensitive to the number of variants available in the library.

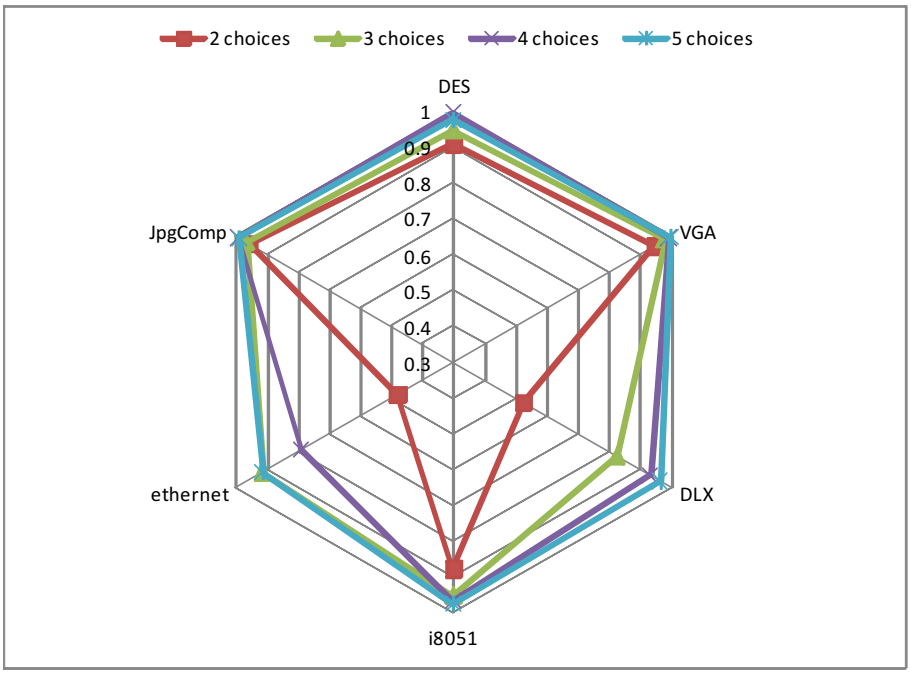

Figure 2.9: Performance improvement achieved when active area sizes are restricted to 2, 3, 4, or 5 discrete variants normalized to that achieved assuming continuous active area sizes. In each experiment, all the available variants are equally spaced between 1X and 2X nominal active area sizes.

In particular, for the benchmark `ethernet`, restricting to only 2 variants (i.e. 1X and 2X only) reduces the performance improvement to only 40% of what is achievable in presence of continuous sizes. As a rule, we observe that having more variants enhances the performance improvement. One exception to this rule is observed for benchmark `des` for which the performance improvement when using 4 choices of active areas (i.e. 1.0X, 1.33X, 1.67X, and 2.0X) is better than having 5 choices (i.e. 1.0X, 1.25X, 1.5X, 1.75x, and 2.0X) of active areas. We believe this is due to different discretization of the solution

space given that the sizes available for 4 choices are not strictly subset of those available for 5 choices. Overall, we observe that having 3 variants (i.e. 1X, 1.5X, and 2X) is a good trade off between performance improvement and cell library sizes.

2.2.9 Results for Experiment Set 2

The above results are very encouraging as the timing improvement achieved is consistent among all different benchmarks. Next, we focus on one benchmark and perform in-depth analysis of the impact of the routing layers, library threshold voltage, and row utilization as outlined in Section 2.2.7. For all these runs, the objective function is to make the circuit as fast as possible. Table 2.6 tabulates some of the representative observation points. The first two column show the row utilization and number of metal layers available to the router. All delay values are presented in adjacent pair of columns for design mapped using low and high V_{th} library variants. The cycle time of original layout after parasitic extraction appears under heading *Orig Timing*. Columns under *Post LP Timing* show the cycle time reported after LP solution when the critical cells are expanded, but before any legalization and ECO routing is done. Columns under *Final Timing* show the final cycle time after legalization and ECO routing. Column *Improvement* tabulates the timing improvement thus achieved (by comparing *Orig Timing* and *Final Timing*). The set of columns under *Cells Legalized* show the number of cells moved due to legalization after the expansion of the critical cells.

The comparison of entries under Column *Post LP Timing* and *Final Timing* shows that the cycle time before and after the legalization and ECO routing stages are usually exactly the same which means that legalization and re-routing did not degrade a non-critical path into critical path. When these entries are not the same, they are very close to each other with a maximum difference of 0.01% in the row utilization range of 0.4-0.7. Similar to the results of experiment set 1, we note that *very* few cells ($\leq$0.7%) were moved during legalization. The total number of nets which need to be ECO routed was found to be under 0.4% of the total nets for all the different configurations. We observe in Table 2.6 that our technique is able to reduce the cycle time of the design by nearly 5.25% averaged over all the benchmarks.

We next discuss the impact of the various design choices such as V_{th} values, number of routing layers used, row utilization of a design on the achieved cycle time reduction using our optimization methodology. Figure 2.10 shows the cycle time reduction achieved using our technique for our benchmark routed in 4 vs 7 metal layers for different row utilization. Our hypothesis was that a design with high routing effort (i.e. with lesser number of metal layers available) would offer less timing improvement due to possible detours during ECO routing of nets ripped by legalizer. The results of the experiments show this to be true most of the times, even though the difference observed is small. We believe that the difference is tiny due to the minuscule number of nets that need to be re-routed owing to constraints in Section 2.2.1. Overall, an average of 5.1% cycle time reduction was achieved over row utilization ratio

between 0.4 and 0.7.

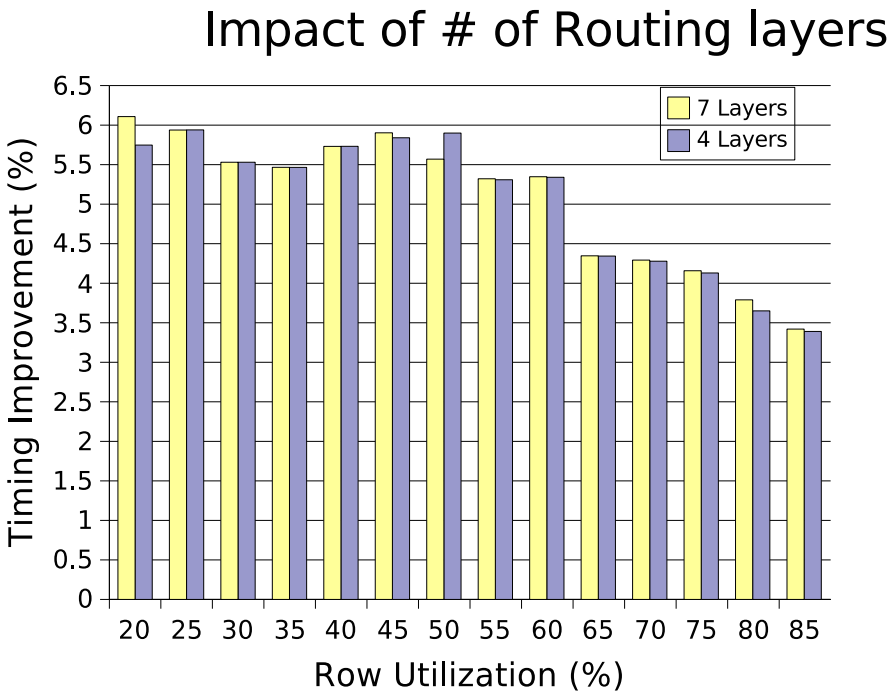

Figure 2.10: Timing Improvement vs Row utilization for Routing using 4 and 7 layers. Benchmark - `wims` High V_{th}.

Figure 2.11 shows the variation of timing improvement for the low threshold design and high threshold design. In general, high threshold voltage designs are slightly more amenable to active sizing expansion based timing optimization method. The reason being that the timing improvement we achieved comes from the decrease in cell delay and high threshold voltage cells have higher ratio of cell delays to interconnect delays. Overall, an average of 5.2% cycle time reduction was achieved for various values of row utilization between 0.4 and 0.7.

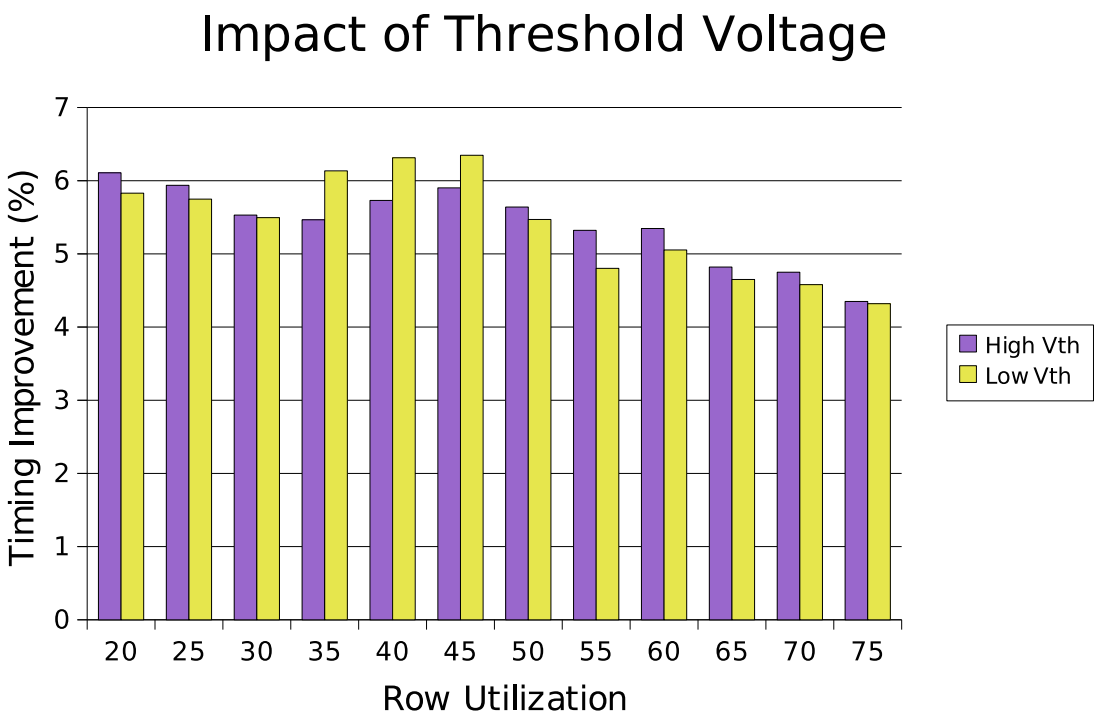

Figure 2.11: Timing improvement vs row utilization for Low and High V_{th} cell library `wims` variants

The amount of whitespace directly impacts our technique, because in essence, our technique consumes whitespace to improve cycle time. Looking at Figure 2.11 and Figure 2.10, we can observe how the timing improvement varies as row utilization is changed for different V_{th} values and routing layers used. As expected, higher row utilization of the design (=lower whitespace), leaves less room for our flow to improve cycle time. Overall, in the practical working range of row utilization of 0.4-0.7, our technique achieved an average cycle time reduction of 5.3%.

2.2.10 Discussions

In this work, we have analyzed the impact of active area sizing on mobility improvement for source/drain Silicon Germanium (S/D SiGe) type devices. An active area stretching aware cell delay model was proposed based on the mobility improvement of the PMOS devices. For the first time, a methodology to exploit this phenomenon inside the conventional design flow is proposed incorporating the active area sizing based optimization during timing closure. A set of constraints was proposed whose adherence results in impeccable predictability of timing improvement after legalization and ECO routing. A legalization algorithm with minimum perturbation to existing cells was also proposed. On several benchmark circuit and on a wide range of design parameters (such as threshold voltages, row utilization, routing congestion), our technique achieves up to 6.3% cycle time reduction which is very attractive considering that the optimization is done at a very late stage in design cycle.

2.3 Stress Aware Repeater Insertion and Gate Sizing

In this section, we will explore the benefits of performing the two most potent timing optimization techniques, i.e. gate sizing and repeater insertion, while concurrently exploiting the mechanical stress dependent mobility of devices.

2.3.1 Stress Aware Cell Delay Model

ITRS roadmap [8] predicts the contact dimension of $56nm$ for the $45nm$ technology node. After accounting for spacer dimensions, we estimate that the typical L_{pp} distance in the $45nm$ process to be around 90-100nm. This estimation also matches the L_{pp} dimension mentioned in [42]. We use the default L_{pp} distance as $90nm$ for the $45nm$ technology node[3]. Consider a nominal PMOS device with $90nm$ L_{pp}. Now, let the AA (i.e. L_{pp}) of this device be increased to K times its original value (i.e. to KL_{pp}). Under such circumstances, let S_K denote the multiplicative increase in the mechanical stress due to increase in active area dimensions. Based on the electrical measurements and simulations data in [42], we plot the increase in mobility, S_K, vs the AA sizing factor K in Figure 2.12.

The different curves above are shown for various recess depths ('R' in figure) and various fractional concentration of Ge in SiGe ('Conc' in figure). From Figure 2.12 we note that increasing recess depth and Ge concentration increases the stress in the channel and that the stress is much more sensitive to Ge concentration than to recess depth. However, these methods are not without their cons: Ge concentration above $35-40\%$ can generate enough lattice misfit [44] to generate dislocations; deeper recess depths increase the interaction between source and drain regions leading to *DIBL* effects. Qualitatively, one can notice the sharp increase in stress as the L_{pp} is sized upto 3 times its

[3]If for a particular fabrication facility this value is different, all that changes is the base value to which we normalize all stress values

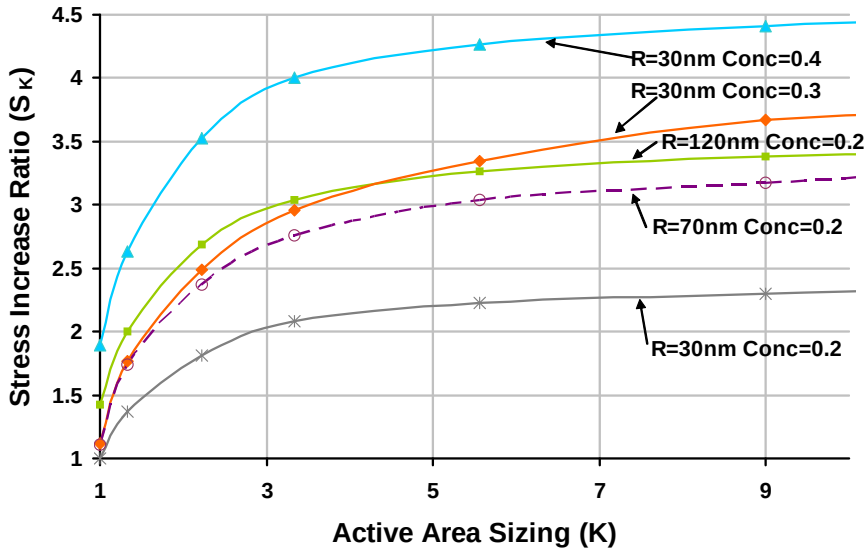

Figure 2.12: Channel stress vs L_{pp} curve. Stress values normalized to the value at nominal L_{pp} of 90nm.

original value (i.e. K=3). The increase in stress is marginal once the value of K goes beyond 5. It is widely accepted that the mobility of the PMOS device is directly proportional ([100] even observed super-linear dependence) to the uniaxial compressive stress[4] in the channel. Therefore the change in mechanical stress, S_K can be mapped on to an equal change in the mobility. From basic device knowledge, we know that the equivalent resistance of a device is

[4]SiGe in S/D region injects uniaxial (only along the channel) stress. STI which surrounds the device area injects bi-axial stress. For bi-axial stress, mobility vs stress curve is in general not linear [88].

given as follows [79].

$$R_{ON} = \frac{V_{DD}}{\mu C_{OX} W \left(V_{GS} - V_T - 0.5 V_{DSAT}\right)} \tag{2.7}$$

Eqn 2.7 implies that the equivalent ON resistance of a device is inversely proportional to its mobility. Hence increasing the mobility (by sizing up L_{pp}) decreases the equivalent resistance of the device. This equivalent resistance can be used for switch level RC analysis of the circuit. Let the resistance of the PMOS device scale down by a multiplicative factor F when the L_{pp} distance is increased by K times. Owing to inverse relationship, F is simply the inverse of S_K of Figure 2.12.

Using the tool GNUplot, we performed curve fitting between F and L_{pp} for the data points corresponding to various Ge concentration and recess depths in Figure 2.12. For each of these cases, the data point fit with high fidelity ($\geq 99.9\%$) to the functional form

$$F = \frac{K}{A \times K + B}$$

where A and B are constants. Putting the boundary condition that K=1 (which means no increase in AA size) would correspond to F=1 (which means no decrease in series PMOS resistance), we get the relationship B=1−A giving

$$F = \frac{K}{A \times K - A + 1} \tag{2.8}$$

From above, we note that A is the only parameter relating F and K. The value of A will depend on the Ge concentration, recess depth and other process

parameters. We will refer to the parameter A as *Single Fitting Parameter* (SFP) here on. A=1 would mean PMOS resistance is independent of AA sizing. A <1 implies *increase* in PMOS resistance with increasing AA size. A >1 implies PMOS resistance *decreases* with increasing AA size. Of course, in our case A >1 and we will use this property to prove the convexity of concurrent Gate and AA sizing later in this section.

For a common value of Ge concentration such as 20% [100] and recess depth of 120nm, Figure 2.13 shows the data points and the fitted curve. The value of *SFP* is found to be 3.4 by curve fitting. For a different SiGe recipe, this constant may be different but it wont affect the general trends of our results. This leads to the final expression as

$$F = \frac{K}{3.4K - 2.4} \tag{2.9}$$

The above model allows predicting reduction in the series resistance of the PMOS transistor as a function of the increase in length of AA (i.e. L_{pp}) around it. We will use the model represented in Eqn 2.8 for theoretical derivation and that in Eqn 2.9 when numerical values are required.

With the above understanding and model, we will develop AA sizing aware cell delay models. For the sake of brevity, we will take an example of 2-input NAND gate but at the end of this section we will provide the final results for other gates such as 3-input NAND, 2-input NOR, 3-input NOR and inverter gate. Consider the toy layout of a 2-input NAND gate shown

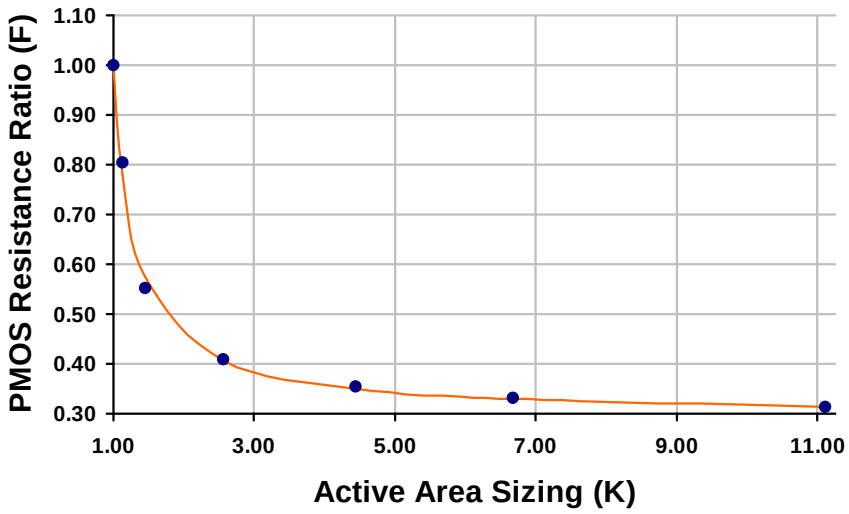

Figure 2.13: Curve fit for decrease in PMOS series resistance vs increase in L_{pp} for 20% Ge and recess depth 120nm

in left half of Figure 2.14 corresponding to the transistor level schematic in top-right. Let symbol R and C denote the resistance and capacitance of a unit width NMOS device. Assuming that the original L_{pp} of the PMOS (as shown in the figure) is $90nm$ and it is increased to K times leading to F times decrease in series PMOS resistance, the RC switch level model of the NAND gate driving a fanout load of hC can be represented as the bottom-right diagram in Figure 2.14. Note that the resistance of the PMOS has been scaled (down) F times ($F \leq 1$) due to AA sizing and the diffusion capacitance of the PMOS has been scaled (up) by K ($K \geq 1$) times to account for larger AA.

Using the notation that the delay of a cell is the average of the fall and

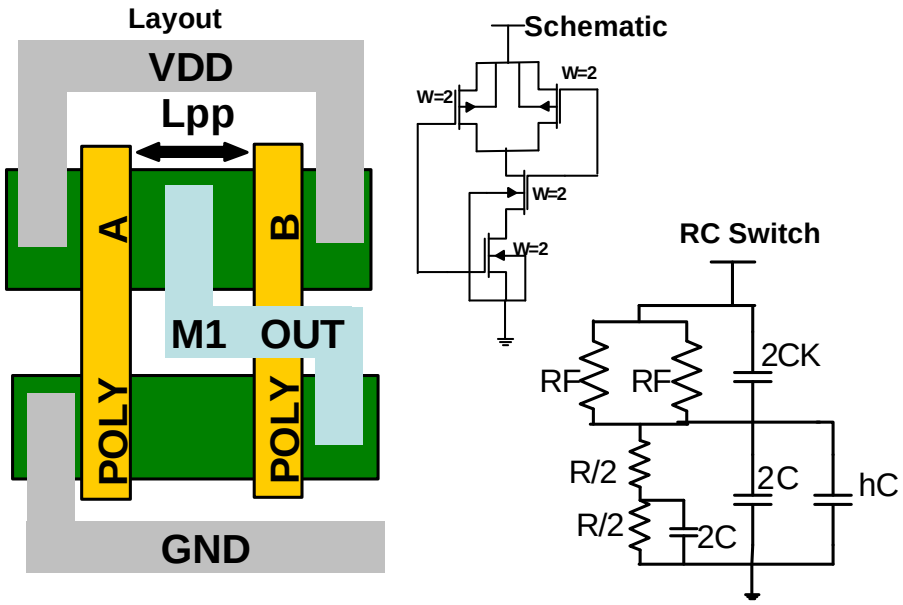

Figure 2.14: Schematic, Layout and RC Switch models for 2-input NAND Gate. Note the impact of AA sizing in switch model

rise times [79], we obtain the AA sizing aware 2-input NAND gate delay as

$$D(K) = RC\,(1+F)\,(1+K+0.5h) + 0.5RC \tag{2.10}$$

Replacing F in terms of SFP from Eqn 2.8 transforms above into

$$D(K) = RC\left(\frac{(A+1)\,K - A + 1}{AK - A + 1}\,(1+K+0.5h) + 0.5\right) \tag{2.11}$$

The nominal delay of the 2-input NAND cell without any AA sizing can be obtained by substituting $K = 1$ in Eqn 2.11 i.e. it it equal to D(1). Define $\Delta D(K)$ as the ratio $D(K)/D(1)$. For a given value of SFP and fanout load h, $\Delta D(K)$ remains only a function of the AA sizing. Consider the case of

$A = 3.4$ which corresponds to the fit derived in Eqn 2.9. Under this scenario, we plot the value of $\Delta D(K)$ vs K in Figure 2.15 for various capacitive load. The term "FO4" refers to fanout of 4 and so on.

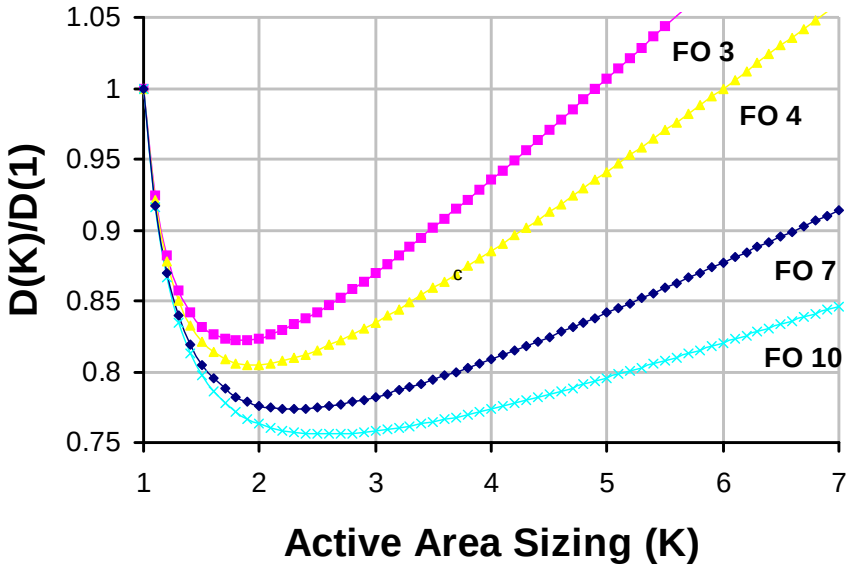

Figure 2.15: Delay decrease (normalized to unstretched cell) vs the extent of AA sizing. Beyond a certain limit, the increase in the AA capacitance overcomes PMOS resistance decrease.

From Figure 2.15, we observe that starting from $K = 1$ (i.e. $L_{pp} =$ 90nm), the delay decreases monotonically until a particular value of K depending upon the fanout load. Let us call this point K_{opt}. Higher is the capacitive load on the cell, more is the value of K_{opt} meaning that highly loaded cells require larger AA size. The delay of the 2-input NAND gate at their optimal delay point is approximately 25% lesser for a fanout of 10 loading and 20% lesser for a fanout of 4 loading. Lets call this delay as D_{opt}. Sizing up the

AA beyond after corresponding K_{opt} does not improve the delay because the increase in AA capacitance outweighs the benefit of reduced PMOS resistance (see Figure 2.14). In fact, if the AA is sized beyond a certain limit, the delay of the gate becomes even bigger than the gate without AA increase. Lets call this point as K_{max} and as an example, K_{max}=6 for 2-input NAND gate with fanout of 4 loading in Figure 2.15. Our analysis can be easily applied to other basic and more complex gates by analyzing its layout and making the equivalent RC switch level network. Based on the method above, we computed K_{opt}, D_{opt} and K_{max} for frequently used gates (NAND, NOR etc) and are shown in Table 2.7.

Let us call Eqn 2.10 as the *characteristic delay equation* for a 2-input NAND gate. On similar lines, the characteristic delay equations of other common gates (after derivation following the steps above) are presented in Table 2.8.

We observe that for the gates in Table 2.7, the delay of the gate can be reduced by around 17% on an average for fanout of 4 loading and by 23% for fanout of 10 loaded gates. Such a significant decrease in delay of the cell can be very useful for high performance designs. We also infer that L_{pp} of each cell needs to be sized up by approximately 75% to 125% of its original value for obtaining the best performance.

2.3.2 Timing Optimization by *AA* Sizing

In this section, after introducing basics of gate sizing and repeater insertion, we will enhance these techniques by performing them aware of *AA* Sizing. As the results will show, these techniques when applied simultaneously with *AA* sizing can push the circuit performance further by 10%. Optimal repeater insertion (ORI) and gate sizing (GS) are the two key techniques routinely used by designers for both microprocessor and ASIC design methodology to perform timing and noise closure. Therefore, combining them seamlessly with *AA* sizing is very beneficial from the practical point of view.

2.3.3 Basics of ORI and GS

In the face of increasing integration and technology scaling, interconnects have been rendered longer and with higher impedance in each generation. ORI problem entails dividing the long interconnect into optimal number of parts and inserting a repeater of optimal gate size so as to reduce total delay over the interconnect. ORI reduces the delay of a global interconnect of length L from a L^2 relationship to linear relationship. This partially alleviates the problem of long global interconnect. Under assumption of same type and structure of repeaters, ORI problem can be solved analytically [79] by obtaining an expression of total delay through the interconnect in terms of the variables such as number of repeaters, gate size of each repeater. This expression can then be minimized w.r.t. the variables to obtain the best configuration of repeaters.

GS problem targets finding the optimal gate size of individual gates in a combinational module so as to minimize an objective such as delay or power under a delay constraint. Consider the channel formed under the gate region (shown as rectangular for simplicity) in Figure 2.16. The channel, source, drain and gate regions marked as C, S, D and G respectively. The target is

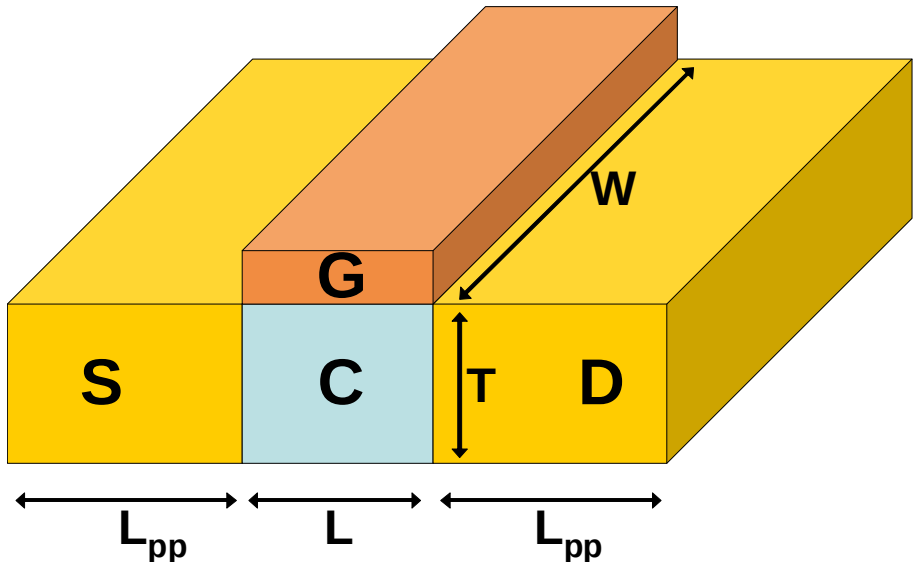

Figure 2.16: Physical structure of a transistor

to flow current from point S in source region to point D in drain region. The resistance offered to the flow of this current, R_{ch}, is given by

$$R_{ch} = \frac{\rho \times L}{W \times T} \tag{2.12}$$

Gate sizing and AA sizing both aim to reduce the R_{ch} to increase the performance of the device. However, there are several major differences in these techniques as highlighted below:

1. Gate sizing entails modulating the dimension W which reduces R_{ch}. On the other hand, AA sizing modifies the dimension L_{AA} which engenders as reduced ρ.

2. Gate sizing changes the capacitive load seen by the driver of the current gate. Thus changing the size of a gate has ripple effect on the gate in its fanin cone. On the contrary, AA sizing changes the capacitance of the source/drain region which is a *local* effect in the sense that the gates in the fanout and fanin cone do not get affected by increasing L_{pp}.

3. Due to the fixed cell height constraint (dimension W in Figure 2.16) in a standard cell type of environment, it may not be feasible to perform gate sizing (changing W) in a continuous way. On the other hand, AA sizing is more amenable/flexible because of there is no such hard constraint on AA size. [5]

2.3.4 Concurrent *AA* Sizing and ORI

Consider an interconnect schematically represented in the top part in Figure 2.17 with L, R_w, C_w as its the total length, per unit length resistance and the per unit length capacitance respectively. Assume that M repeaters are inserted in it, i.e. each repeater drives an interconnect of length L/M. Our aim is to achieve fastest delay[6] from the driver to the sink of the interconnect.

[5] We note that to reduce cell library size, both gate sizing and AA sizing would need to be discretized, but gate sizing is still *more* discontinuous due to hard constraint on cell height

[6] For objective other than delay minimization, similar approach can be followed by using our switch level RC model

If S and K represent the gate and AA sizing factors for each of these repeaters, its RC switch level model is as shown in the lower part of Figure 2.17. The resistance reduction factor F is shown in the expression of series resistance of the PMOS transistor. The interconnect segment is modeled as a π network.

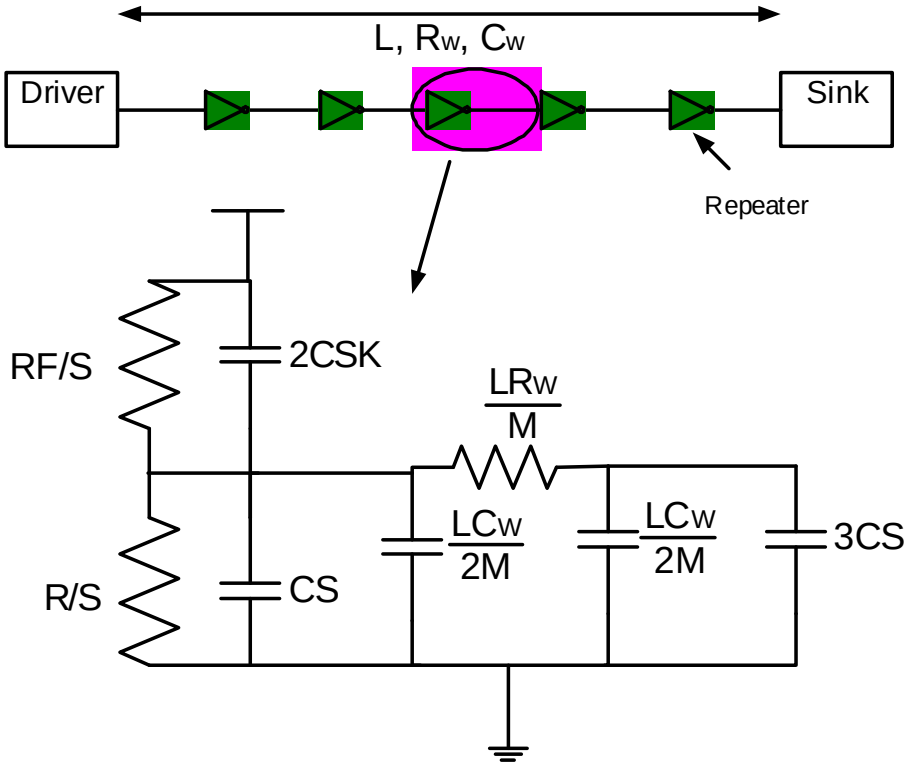

Figure 2.17: Schematic of a long interconnect with repeaters inserted (top). Switch level RC model for the repeater considering π model of interconnect (bottom)

The switched capacitance and Elmore delay of *one stage* as well as the delay of the whole inverter chain, D_{tot} can be respectively computed from

Figure 2.17 as

$$C_{stage} = 2CSK + 4CS + LC_w/M \tag{2.13}$$

$$D_{stage} = \frac{(F+1)\,RC_{stage}}{2S} + \frac{R_w L}{M}\left(\frac{LC_w}{2M} + 3CS\right) \tag{2.14}$$

$$D_{tot} = M \times D_{stage} \tag{2.15}$$

2.3.4.1 Optimizing Delay

To minimize D_{tot}, its partial derivatives with respect to gate size, number of stages and AA sizing should be 0. Solving for $\frac{\partial D_{tot}}{\partial S} = \frac{\partial D_{tot}}{\partial M} = 0$ and after some algebra, we obtained the following results.

$$M_{opt} = L\sqrt{\frac{C_w R_w}{2RC(F+1)(K+2)}} \tag{2.16}$$

$$S_{opt} = \sqrt{\frac{RC_w(1+F)}{6R_w C}} \tag{2.17}$$

where M_{opt} and S_{opt} represent the optimal number of repeaters and sizing of each of them. As the expressions of S_{opt} and M_{opt} are independent of each other, they can be independently set to their optimal value. Apart from the fixed circuit parameters (i.e. R, C, R_w, C_w and L), M_{opt} and S_{opt} depend on the value of K (the amount by which the AA is sized up)[7]. This is the major difference between traditional ORI and ORI considering the freedom of AA sizing.

As the AA is made bigger (i.e. K increases), the value of series resistance multiplicative factor F decreases (see Eqn 2.9). Thus, the value of M_{opt}

[7]Since the value of F depends only on K as per Equation 2.8

is affected by two reverse trends: $K + 2$ term increases whereas $1 + F$ term decreases. On the other hand, the trend of S_{opt} is straightforward: as the AA is made bigger, $1 + F$ term decreases thus decreasing the optimal gate sizing value S_{opt}. The reduction of S_{opt} with increase in AA sizing is intuitive because as K increases, more and more performance increase comes from AA increase thus requiring increasingly smaller gate sizes. Figure 2.18 shows the value of M_{opt} and S_{opt} as a function of K normalized to their value without any AA sizing up i.e. at K=1.

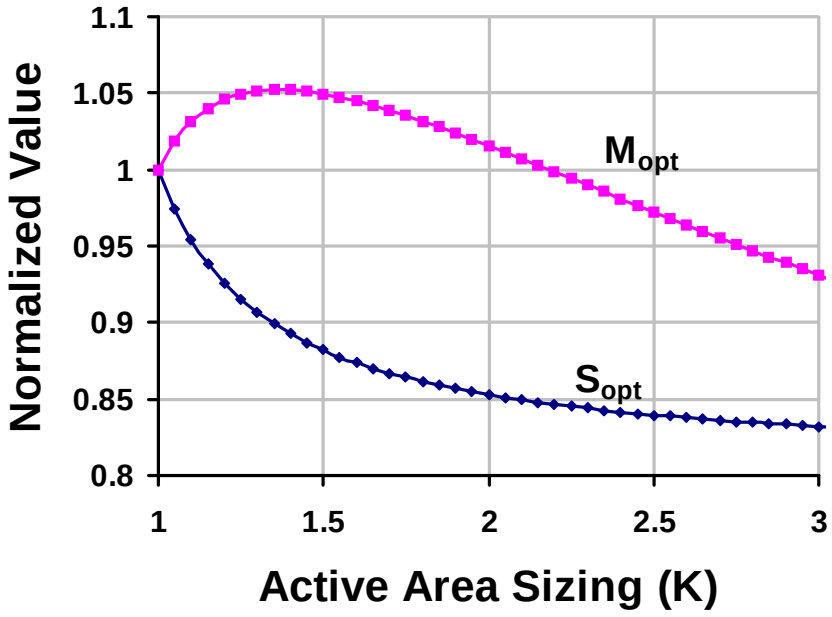

Figure 2.18: Optimal repeater size (S_{opt}) and number of repeaters (M_{opt}) as a function of AA sizing (K).

From Figure 2.18 we observe that for a gate with slightly larger AA

(say 25%, corresponding to K=1.25) the number of repeaters for optimal delay is more ($\sim 5\%$) than its counterpart without AA sizing. On substituting the optimal number of stages (M_{opt}) from Eqn 2.16 and optimal inverter size (S_{opt}) from Eqn 2.17 into Eqn 2.15, we obtain D_{tot} as

$$D_{tot} = \sqrt{2L^2RCR_wC_w} \times \sqrt{1+F} \times \left(\sqrt{3} + \sqrt{2+K}\right) \tag{2.18}$$

At this stage, the only unknown independent variable in the above equation is K whose optimal value can be found by solving for $\frac{\partial D_{tot}}{\partial K} = 0$. Using the generic expression for F in terms of K from Eqn 2.8 into Eqn 2.18 results in an unwieldy ninth order expression for K which does not provide any insight. Therefore, we computed the value of K_{opt} for a range of numerical values of the *SFP* A. Qualitatively, higher is the value of the *SFP*, sharper is the decrease in series resistance of the PMOS, thus larger is the value of AA sizing before the increase in AA capacitance starts overcoming the benefit of reduced series resistance. This trend is clear in Figure 2.19 which shows the value of K_{opt} as a function of the *SFP* using Eqn 2.9. As expected, the optimum value of AA size, K_{opt} increases as the *SFP* increases. Further, from Figure 2.19 we observe that once the value of *SFP* is more than a certain value (approximately 3.5 in Figure 2.19), the dependence of K_{opt} on *SFP* becomes very weak. For very small values of *SFP* ($A \leq 1.3$), the reduction in PMOS resistance is much lesser than the increase in capacitive loading, thus the optimal value of K is equal to 1 i.e. the cell should *not* be made with larger AA at all. As long as the functional fit of the form Eqn 2.8 holds true, the optimal extent of AA sizing, K_{opt} can be read off Figure 2.19.

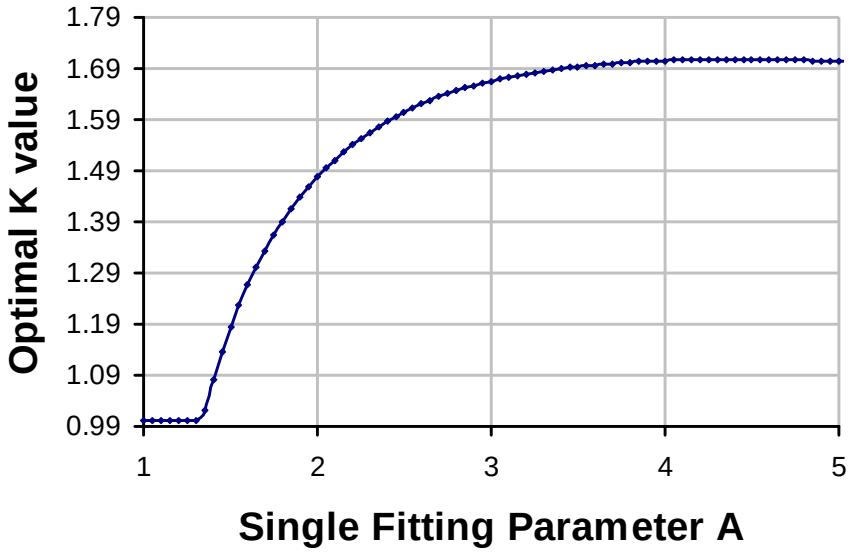

Figure 2.19: Dependence of optimal AA sizing factor K_{opt} on the SFP A. Larger A can support larger AA increase.

At this stage, we are ready to solve the AA sizing aware ORI problem completely. The result of this problem includes: a) AA sizing of each repeater, b) gate sizing of each repeater, c) number of repeaters, d) resultant delay of the interconnect, e) power consumption. For the ease of understanding the impact AA awareness had on repeater insertion problem we will present the results normalized to the case of ORI without any AA sizing.

- For the case of numerical fit obtained in Eqn 2.9 the value of K_{opt} can be seen from the Figure 2.19 as

$$K_{opt}|_{A=3.4} = 1.69 \tag{2.19}$$

This means that the active area of each inverter cell should be increased

to 1.69 times its original value for achieving shortest delay. This value of K is used to get other results.

- Using K=1.69 into Eqn 2.16, the optimal number of repeaters, M_{opt}, is 4% more than its value without any AA increase. Placement tools should consider this while whitespace allocation.

- Eqn 2.17 results in a value of S_{opt} which is 13% *smaller* than its nominal value without AA size increase. This implies that inverters with smaller width, but larger AA are required for least delay.

- In Figure 2.20 we plot the fastest delay through the interconnect (normalized to least delay without AA sizing) as a function of sizing. At $K = 1.69$, the delay through the long interconnect is reduced by approximately 9%.

- The dynamic power for *each* repeater is given as $C_L V_{DD}^2 f$ where C_L is the switched capacitance as given in Eqn 2.13 and f, the frequency, can be expressed as $\frac{1}{D_{rptr}}$ where D_{rptr} is the delay through the repeater. The ratio of dynamic power in the complete repeater chain incurred due to increased AA size can be calculated as

$$\frac{P_{stretch}}{P_{nominal}} = \frac{[M \times 1/D_{rptr} \times C_L]_{K=1.69}}{[M \times 1/D_{rptr} \times C_L]_{K=1.00}} \tag{2.20}$$

Substituting Eqn 2.17 in Eqn 2.13 and using it along with Eqn 2.16, Eqn 2.18 and after some algebra, the above equation condenses into an elegant

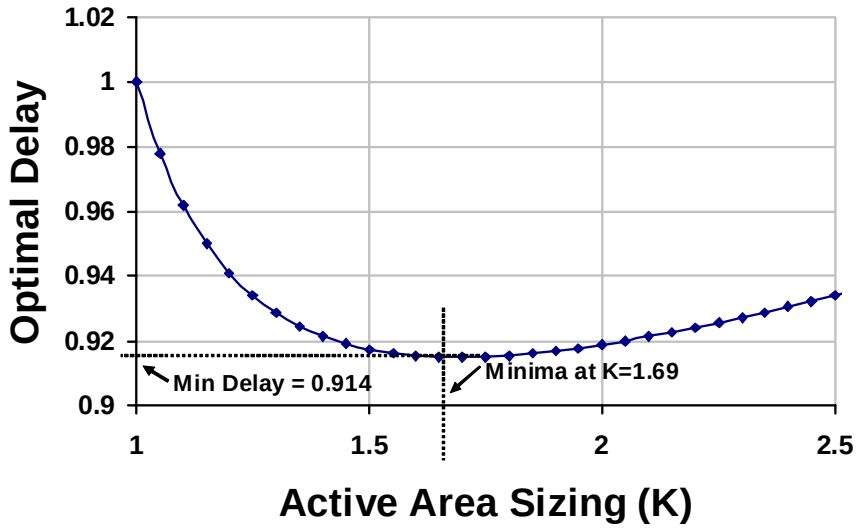

Figure 2.20: Normalized delay through the repeater chain vs the AA sizing of each repeater

simple relation below

$$\frac{P_{stretch}}{P_{nominal}} = \frac{\left(1/\sqrt{1+F}\right)|_{K=1.69}}{\left(1/\sqrt{1+F}\right)|_{K=1.00}} = 1.15 \tag{2.21}$$

Thus, the total dynamic power dissipation of the repeater chain increases by 15% due to AA sizing. This increase comes due to three components: a) increased number of repeaters (4%), b) increased AA capacitance (2%) and c) increased frequency of operation(9%).

Summary: Table 2.9 summarizes the results of AA sizing aware ORI for minimizing delay of long interconnect. The minimum achievable delay, optimal AA size, number of repeaters, repeater sizing, minimum achievable delay and dynamic power dissipation is reported where each of these is normalized

to the case without the flexibility of exploiting AA sizing. These parameters uniquely define the solution of ORI.

2.3.4.2 Minimizing Repeater Number

AA aware repeater insertion can be used to reduce the repeaters inserted in the iso-delay case w.r.t. repeater insertion without AA sizing. Let us assume that the delay through an interconnect after performing ORI alone (i.e. without AA sizing) is D_{ORI}. If ORI and AA sizing was performed concurrent as in the previous section, let the resultant delay be D_{ORI+DS} (note that $D_{ORI+DS} < D_{ORI}$) . We want to find out by how much can we reduce the number of repeaters in the second case so that the sub-optimal delay D_{ORI+DS} becomes just equal to D_{ORI}. This scenario is of importance for the global interconnects which are not the most critical ones. Since repeaters are power hungry elements and their insertion cause substantial difficulty in physical synthesis, reducing their number is advantageous.

Solving ORI without AA size increase (by substituting K=1 in Eqn 2.16 and Eqn 2.17 and putting these values back in Eqn 2.14) yields the result

$$D_{ORI} = 4L\sqrt{3RCR_wC_w} \tag{2.22}$$

Now, consider the case of ORI concurrently with AA sizing. We can put S=S_{opt} and F corresponding to K=1.69 but instead of assigning M=M_{opt}, let M=δM_{opt} where $\delta \in (0, 1]$. In such a case, it can be shown that

$$D_{ORI+DS} = L\sqrt{RCR_wC_w} \times \left(3 + \frac{2}{3\delta} + \frac{2\delta}{3}\right) \tag{2.23}$$

Summary: Solving Eqn 2.22 and Eqn 2.23, we obtain $\delta = 0.55$. This means that by using concurrent ORI and AA sizing, we can save nearly half (45%) of repeaters without sacrificing any performance w.r.t. ORI without AA size increase. This is a very interesting result and can have substantial impact due to aforementioned benefits of reducing repeater count.

2.3.5 Concurrent Gate and AA Sizing

Gate sizing under Elmore delay is a well under-stood problem and can be solved using convex solvers by exploiting the property of a class of function called *posynomials*, that allows conversion of the optimization problem into a convex programming problem with a straightforward transformation [25]. A larger gate is more effective at driving big capacitive loads at the cost of presenting higher load to its input gate. Based on the understanding of comparison between gate and AA sizing in Section 2.3.3, an obvious question that springs up is about the possibility of combining these two techniques for faster circuits. In section 2.3.4, we analytically combined ORI and AA sizing. That problem was amenable to analytical solution due to the fact that all repeaters were similar to each other. In a general circuit, there can be several different cells, multiple fanin and fanout gates etc and trying to analytically solve for minimum delay is impossible. We thus formulate the Concurrent Gate and Active area Sizing (CGAS) problem as

2.3.5.1 CGAS Formulation

> CGAS: Given a multilevel circuit to be fabricated with SiGe S/D technology, perform concurrent gate and AA sizing (CGAS) to minimize an arbitrary convex[8] cost function

Consider Gate 1 in Figure 2.21 with tuple {S1, C1, K1} which represents its gate sizing, gate capacitance per input pin and AA sizing respectively. Gate 1 drives Gate 2 and 3 associated with their tuples {S2, C2, K2} and {S3, C3, K3}. For the sake of clarity, consider for now that Gate 1 is an inverter whose AA sizing aware RC switch level model is given in the inset in Figure 2.21. $a1$ through $a4$ are the signal arrival times at the different locations in the figure. The delay of Gate 1 can be written as

$$D_1 = RC\frac{(1+F_1)}{2}\left(1 + 2K_1 + \frac{(S_2C_2 + S_3C_3))}{S_1}\right) \tag{2.24}$$

where F_1 depends on K_1 as in Eqn 2.8. Note that $S_2C_2 + S_3C_3$ can also be written as $\sum_{m\in FO_1} C_m S_m$, where FO_i represents the fanout of gate i. Using Figure 2.21, we can write the following equations for the arrival times.

$$at_0 + D_1 \leq at_2$$

$$at_1 + D_1 \leq at_2$$

where D_1 is given in Eqn 2.24. Now consider the case when Gate 1 is *not* an inverter but some other logic gate. Under such a scenario we can write a

[8] As we will prove later, all the constraints of CGAS are convex therefore having a convex cost function guarantees the problem to be convex and thus can be solved optimally.

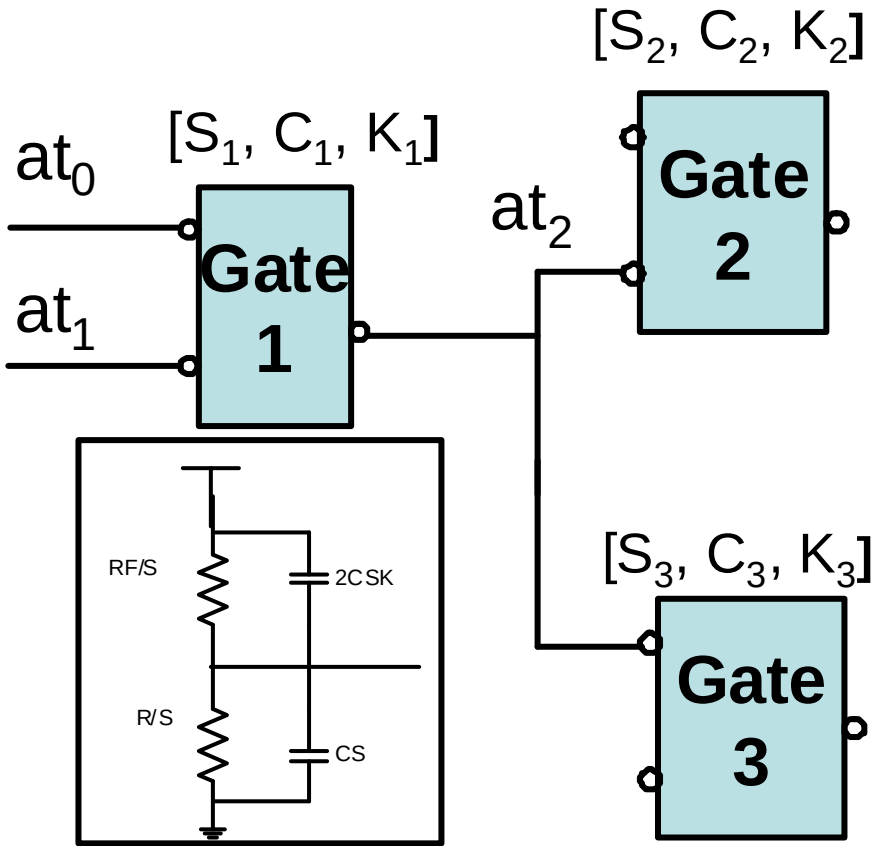

Figure 2.21: A part of logic circuit under consideration. Gate 1 drives Gate 2 and 3.

generic expression (see particular examples in Table 2.8) of the delay of gate i by generalizing Eqn 2.24 as

$$D_i = RC\frac{(1+F_i)}{2}\left(a + bK_i + \sum_{m \in FO_i} \frac{c_m S_m}{S_i}\right) \tag{2.25}$$

where a, b and various $c_m \in \mathbf{R}^+$.

Let us now consider an unoptimized multi-level circuit in which I, O, L represent the set of input pins, output pins and internal logic gates respectively. For each gate i, let at_i, S_i, K_i, FI_i, FO_i denote its arrival time at its output, gate sizing, AA sizing factor, fanin gates and fanout gates respectively. $CGAS$ can now be written as a mathematical optimization program as

$$\begin{aligned} &Minimize : Delay \\ &Subject\ To : \\ &\quad at_j + D_i \leq at_i \quad \forall i \in L, \forall j \in FI_i \\ &\quad at_i = 0 \qquad \forall i \in \{I\} \\ &\quad Delay > at_i \quad \forall i \in \{O\} \\ &\quad at_i > 0 \qquad \forall i \in \{L \cup O\} \\ &\quad S_i, K_i > 1 \qquad \forall i \in \{L \cup O\} \end{aligned} \tag{2.26}$$

The dummy variable $Delay$ represents the largest of the arrival times among various output pins of the circuit in the third set of constraints. Therefore, minimizing $Delay$ is equivalent to making the circuit as fast as possible. Though the above formulation is targeted for fastest possible circuit implementation, we note that any other traditional objectives can be handled easily too. For example, for optimizing power consumption, we can perform AA minimization under a delay constraint by adding one constraint for the required arrival time of the circuit.

2.3.5.2 Convexity of CGAS

The objective function as well as all constraints except the first set of constraints in the *CGAS* are convex by observation. We focus on the first set of constraints and show that it is posynomial. Using Eqn 2.25 for the value of D_i and Eqn 2.8 for the value of F, each of the first set of delay constraints can be written as

$$\left(1+\frac{K_i}{AK_i-A+1}\right)\times\left(a_i+b_iK_i+\sum_{m\in FO_i}\frac{c_mS_m}{S_i}\right)\leq\frac{2\left(at_i-at_j\right)}{RC} \tag{2.27}$$

such that A, a, b, c_m, S_m and $S_i \in \mathbf{R}^+$. Substituting AK_i-A+1=T_i, RHS by Δat and S_m/S_i by S_{mi}, the above can be written as

$$\left(1+\frac{1}{A}+T_i\left(1-\frac{1}{A}\right)\right)\times\left(a+\frac{bT_i}{a}+b\left(1-\frac{1}{a}\right)+\sum_{m\in FO_i}c_mS_{mi}\right)\leq\Delta at \tag{2.28}$$

On cross multiplying, the *LHS* of the above equation can be separated in terms T_i and various S_{mi} as

$$MNT_i^2+(NL+PM)\,T_i+MT_i\sum_{m\in FO_i}c_mS_{mi}+L\sum_{m\in FO_i}c_mS_{mi} \tag{2.29}$$

where, L=$1+\frac{1}{A}$, M=$1-\frac{1}{A}$, N=$\frac{b}{a}$ and P=$a+b\left(1-\frac{1}{a}\right)$. L, N and M (since A>1 for Eqn 2.9) $\in \mathbf{R}^+$ by inspection. Using this, the coefficients for T_i^2, $S_{mi}T_i$ and $S_{mi} \in \mathbf{R}^+$. To prove that the above expression is posynomial we need to prove that the coefficient of $T_i \in \mathbf{R}^+$. Since the value of a can be less than 1 (e.g. INV and NOR gate in Table 2.8) this cannot be done simply

by inspection. Thus, we simplify the coefficient of T_i

$$NL + PM = \frac{b}{a}\left(1 + \frac{1}{A}\right) + \left(1 - \frac{1}{A}\right)\left(a + b\left(1 - \frac{1}{a}\right)\right)$$
$$= \frac{1}{A}(A-1)(a+b) + \frac{2b}{a}$$

From the above form (since A>1), it is clear that the coefficient of T_i also $\in \mathbf{R}^+$ and thus each constraint in *CGAS* is the form of a posynomial. Under a elementary variable transformation using exponential functions, this constraint can be mapped into a convex constraint [25]. Therefore problem *CGAS* is convex and can be solved with existing convex program solvers optimally.

2.3.5.3 Power Considerations

Though increasing *AA* size reduces the PMOS resistance, it overall increases the *AA* capacitance which impacts the dynamic power dissipation of the design. To quantify the impact of our technique CGAS has on dynamic power (w.r.t. to technique GS), we computed the total switched capacitance for gate sizing, C_{GS} and concurrent gate and *AA* sizing , C_{CGAS}, as

$$C_{CGAS} = \sum_{i \in L}\left(A_i + B_i k_i + \sum_{m \in fo_i} \frac{C_m s_m}{s_i}\right) \tag{2.30}$$

C_{GS} can by found evaluating $C_{CGAS}|_{\forall k_i = 1.00}$. If CGAS can result in a circuit which runs T times faster than the circuit realized with conventional GS, the increase in dynamic power consumption of the new circuit would then be given as

$$\Delta P_{dyn} = \frac{T \times C_{CGAS}}{C_{GS}} \tag{2.31}$$

Note that since the underlying canonical circuit and logic structure remains the same for the two differently sized circuit realizations, they will have exactly same switching factors for each node. Therefore, we can safely ignore the impact of scaling each capacitance value by corresponding switching factor. Given the solution of of *GS* and *CGAS*, it is possible to find out the value of ΔP_{dyn}. We followed a 2-step procedure to calculate C_{CGAS} and C_{GS} . In the first step, *CGAS* program of Eqn 2.26 was solved to achieve minimum possible delay. Let this minimum delay be D_{min}^{CGAS}. In the second step, the objective function in Eqn 2.26 was modified to minimize C_{CGAS} under the constraint that the timing of the resultant circuit is at most D_{min}^{CGAS}. This way, the circuit realization with least switching capacitance subject to timing constraints was obtained. In the same fashion, the minimum dynamic power for circuit realization obtained by traditional GS was also found out. The obtained capacitance values for these two cases determine ΔP_{dyn}.

2.3.6 Experimental Setup and Results

We solved the *CGAS* problem on a variety of benchmarks from IWLS 1991 LGSynth suite [7]. These benchmarks are implementations of a soup of logic blocks and we believe that such mixture of benchmarks can counter structural bias present in them. The original *blif* format of the benchmarks were optimized and technology mapped using SIS program [86]. For the sake of simplicity we restricted our library to a set of 3 gates: 2-input NAND, 2-input NOR and an Inverter. The characteristic delay equations of these gates

are in Table 2.8. We note that typically a cell library can have several other functionally unique cells and a typical industrial application of our technique will require designer to extend Table 2.8 accordingly for cells such as XOR, AOI with similar results. A C++ program was used to write out the objective and constraints for the tool AMPL [45] which was coupled with the back-end solver MOSEK [10] that solves the convex program using interior point optimization method. Table 2.10 shows the comparison of efficacy of Gate Sizing (GS) alone and concurrent Gate and Active area Sizing (CGAS) for timing optimization. Column "# Gates" shows the total number of gates in each benchmark. The timing achieved with the respective techniques is shown in Columns "Delay" with the delay enhancement of *CGAS* over *GS* shown in column "Improv". Column "ΔCap" and "ΔP_{dyn}" show the increase in switching capacitance and dynamic power of solution of *CGAS* over *GS*.

The results in Table 2.10 are very encouraging. We observe that *CGAS* can push the envelop of performance by more than 10% over and above the optimal delay value achievable by traditional gate sizing. For high performance designs, this is very attractive. A 10.08% decrease in design cycle time corresponds to more than 11.2% increase in the chip operating frequency. On comparing the increase in switched capacitance of each benchmark, we found that the capacitance increase is very modest at under 1%. This number is very much dependent on the structure of the benchmark and lies in the wide range of 0.2% to 3.2% for different benchmarks. Results show that the dynamic power increase due to *CGAS* is around 11% on an average. Out of it, nearly

10% increase is simply because the design can switch at a higher frequency of operation. The rest 1% increase is due to capacitance increase for the larger AA region. It should be noted that in the absence of $CGAS$, it is impossible to improve performance beyond the optimal solution of traditional gate sizing (without changing V_{DD} or reducing V_{th}). For each of the benchmarks in Table 2.10, the convex solver took less than a minute to attain the optimal solution.

2.3.7 Discussions

AA sizing for SiGe S/D type devices opens up an exciting dimension for optimizing their performance. Gates with high fanout loads are most benefitted by sizing up AA - this could be useful as the interconnect capacitance increases in future technology nodes. Enlarging active area can have mixed impact on the recommendation of using unidirectional poly in layout. For cells whose n-well is larger than p-well (like NAND gate), increasing the active area of PMOS can cause longer wrong way poly routing. On the other hand, for cells which have *smaller* n-well than p-well (like NOR gates), active area sizing of PMOS can actually bring the layout close to unidirectional case. In our results, we observed dynamic power increase of the order of 11%. We believe that for high performance design applications, it is a fair price to pay for getting 10% decrease in design cycle time. Use of AA sizing alters footprints of the cells and the design flow should be modified to account for it.c

Design	Orig Timing a	Post LP Timing b	Final Timing c	Post LP Improv d	Final Improv e	Δ Improv $d-e$	# cells critical g	# cells stretch h	# cells Move i	ΔPower Dynamic j	ΔPower Leak k	ECO Nets l
des	6.153	5.569	5.570	4.74%	4.74%	0%	13092	48	5	0.39%	2.53%	0.09%
vga	2.081	1.792	1.799	6.94%	6.77%	0.17%	116	34	23	0.61%	5.24%	1.91%
dlx	3.343	2.922	2.927	6.29%	6.22%	0.07%	548	172	104	0.94%	7.33%	3.41%
i8051	11.265	10.248	10.249	4.51%	4.51%	0%	131	62	95	0.89%	4.97%	2.79%
ethernet	28.343	24.463	24.507	6.84%	6.76%	0.08%	2406	299	92	0.34%	2.88&	0.36%
JpgComp	7.238	6.425	6.447	5.61%	5.46%	0.15%	273	80	296	0.70%	4.86%	5.46%
Avg.				5.82%	5.74%	0.06%		0.42%	0.37%	0.58%	4.63%	2.36%

Table 2.3: Results of our flow to achieve fastest possible circuit. See accompanying text for columns' description.

Design	Orig Timing	# Crit Cells	T_{tgt} = 99% Orig.			T_{tgt} = 98% Orig.		
			a	b	c	a	b	c
des	6.153	13092	1	0	0.3m%	1	1	0.6m%
vga	2.081	116	3	3	45m%	4	3	91m%
dlx	3.343	548	1	0	3m%	1	0	6m%
i8051	11.26	131	6	7	37m%	14	26	132m%
ethernet	28.34	2406	2	0	0.2m%	4	0	0.8m%
JpgComp	7.238	273	1	6	2m%	4	19	14m%

Table 2.4: Results for achieving target timing (T_{tgt}) with least power increase. T_{tgt} varied through 99% and 98% of original timing. Columns a, b, and c show no. of cells stretched, no. of cells moved during legalization, and dynamic power increase (in millipercent = 0.001%) for each T_{tgt}. Entries marked NA means the corresponding T_{tgt} was not achievable.

Design	T_{tgt} = 97% Orig.			T_{tgt} = 96% Orig.			T_{tgt} = 95% Orig.		
	a	b	c	a	b	c	a	b	c
des	2	2	1m%	4	3	1.3m%	NA	NA	NA
vga	6	4	139m%	8	4	188m%	12	15	244m%
dlx	8	5	23m%	20	5	90m%	39	15	200m%
i8051	31	40	340m%	47	67	676m%	NA	NA	NA
ethernet	6	0	2m%	7	0	3m%	7	0	5m%
JpgComp	11	27	52m%	27	54	143m%	56	151	338m%

Table 2.5: Results for achieving target timing of 97%, 96%, and 95% of original timing, in continuation of Table 2.4. Entries marked NA means the corresponding T_{tgt} was not achievable.

Design	Row Util	Rout Layr	Orig. Timing(ns)		Post LP Timing(ns)		Final Timing(ns)		Improvement (%)		No. Cells Legalized	
			LVT	HVT	LVT	HVT	LVT	HVT	LVT	HVT	LVT	HVT
wims	0.20	4	9.879	40.171	8.727	35.553	8.727	35.553	5.83 %	5.75 %	78	88
wims	0.20	7	9.883	40.180	8.731	35.271	8.731	35.271	5.83%	6.11%	80	89
wims	0.30	4	9.389	32.889	8.352	29.260	8.352	29.260	5.50%	5.53%	83	79
wims	0.30	7	9.379	32.811	8.349	29.182	8.349	29.182	5.52%	5.53%	79	80
wims	0.40	4	9.257	31.923	8.089	28.261	8.088	28.264	6.31%	5.73%%	79	74
wims	0.40	7	9.241	31.918	8.074	28.257	8.074	28.260	6.32%	5.73%	80	78
wims	0.50	4	9.060	32.110	8.068	28.322	8.068	28.322	5.47%	5.90%	76	69
wims	0.50	7	9.062	31.423	8.071	28.423	8.071	28.423	5.47%	5.68%	84	69
wims	0.60	4	8.865	31.242	7.969	26.452	7.969	26.455	5.05%	4.95%	67	78
wims	0.60	7	8.862	29.066	7.969	25.954	7.969	25.958	5.05%	5.35%	77	73
wims	0.70	4	8.354	31.734	7.584	29.019	7.580	29.019	4.65%	4.28%	80	69
wims	0.70	7	8.343	31.703	7.567	28.981	7.567	28.981	4.63%	4.29%	78	78
wims	0.80	4	8.161	30.823	7.558	27.846	7.558	27.842	3.67%	4.12%	72	84
wims	0.80	7	8.106	29.672	7.511	26.582	7.573	26.582	3.70%	5.20%	81	76
Avg.									5.21%	5.29%	78	77

Table 2.6: Results for achieving fastest possible circuit for benchmark `wims` for different row utilization, threshold voltages (LVT=Low V_{th}, HVT=High V_{th}), and routing layers.

	Fanout 4			Fanout 10		
Name	K_{opt}	D_{opt}	K_{max}	K_{opt}	D_{opt}	K_{max}
INV	1.78	0.82	4.63	2.31	0.76	9.54
2-NAND	1.97	0.80	6.00	2.57	0.75	12.45
3-NAND	1.72	0.84	4.22	2.20	0.78	8.31
2-NOR	1.66	0.83	3.79	2.11	0.78	7.46
3-NOR	1.62	0.84	3.55	2.03	0.78	6.66
AVERAGE	1.75	0.83	4.43	2.24	0.77	8.88

Table 2.7: Optimal AA sizing, optimal delay (normalized to cell without AA sizing) and maximum sizing for common gates

Name	Equation
INV	$RC\,(F+1)\,(K+0.5h+0.5)$
2-NAND	$RC\,((F+1)\,(K+0.5h+1)+0.5)$
3-NAND	$RC\,((F+1)\,(2K+0.5h+1.5)+1.5)$
2-NOR	$RC\,((F+1)\,(2K+0.5h+0.5)+FK)$
3-NOR	$RC\,((F+1)\,(3K+0.5h+1)+3FK)$

Table 2.8: AA Sizing aware Elmore Delay Equations for some common gates

$\mathbf{D_{tot}}$	**AA Sizing**	**Gate Sizing**	**# Rptrs**	**Power**
0.91	1.69	0.87	1.04	1.15

Table 2.9: Solution of Active Area Sizing aware ORI Problem. (normalized w.r.t. the values without increasing AA size)

Design	Gates	Delay (times RC)		Imprv	ΔCap	ΔP_{dyn}
		GS	CGAS	%	%	%
C6288	3316	1320.65	1175.15	11.01	3.20	14.67
C880	502	340.83	309.19	9.03	0.38	9.44
frg1	149	178.44	159.34	10.70	0.19	10.82
k2	1163	323.06	295.13	8.65	0.49	9.19
C7552	2581	734.07	687.49	6.40	0.43	6.85
large	481	262.51	236.90	9.75	0.30	10.08
vda	628	222.38	199.76	10.17	0.57	10.80
des	3759	270.87	233.22	13.89	1.34	15.43
C5315	2007	449.92	400.20	11.05	1.54	12.78
Avg.				10.08	0.94	11.17

Table 2.10: Performance Improvement and Dynamic power increase for various benchmarks

Chapter 3

Controlling Negative Bias Temperature Instability (NBTI) Aging of Nanometer ICs

3.1 Background of NBTI Aging

NBTI (Negative Bias Temperature Instability) has emerged as the primary PMOS failure mechanism[23] for advanced sub-65nm VLSI technology. Several technology trends have caused NBTI degradation to occur at even higher rates in newer technology nodes. These trends include reduction in gate oxide thickness which increases electric field across the gate oxide, and increased use of nitride to reduce Boron penetration [17]. Due to these trends, NBTI dictates the lifetime of the device as compared to other reliability issues such as hot carrier injection, time dependence dielectric breakdown etc. NBTI causes slow threshold voltage (V_{TH}) degradation (i.e. increase) of PMOS device consequently reducing its drive current and performance over time. Over a period of 10 years, the V_{TH} of the PMOS device can increase by up to 50mV[98] causing timing violation and functional failures. As feature sizes shrink, NBTI effects will worsen exponentially due to higher operating temperatures. The instability of PMOS transistor parameters (e.g., threshold voltage, transconductance, saturation current, etc.) under negative (inversion) bias and relatively high temperature has been known to be a reliability

concern since the 1970s, and modeling effort to understand this is also just as old [16]. Though similar effect also occurs in NMOS devices under positive bias [39], the degradation in PMOS devices is much higher than in NMOS devices.

The shift in V_{TH} of the PMOS undergoing NBTI is due to the generation of interface traps under negative gate-to-source bias. These interface traps are the result of breaking of weak Si–H bonds which are formed due to crystal mismatch at the channel-gate interface[66]. Figure 3.1 from [16] illustrates the complete process of interface trap generation. The breaking of Si–H bonds at the Si–SiO_2 interface creates Si^+ (interface traps) and H atom. Initial interface-trap generation rate depends on Si–H bond dissociation (reaction), while the steady state rate depends on H removal (diffusion) rates. The presence of Si^+ at the surface requires larger gate voltage for channel inversion. This is the reason V_{TH} of a PMOS device increases due to NBTI. The negatively biased duration of the PMOS is said to be *stress* stage. Removal of the negative gate-to-source bias helps in annealing *some* of these interface traps, thereby leading to partial recovery. This phase is known as *recovery stage*. As the internal nodes in a circuit switch during regular operation, each PMOS device experiences a sequence of stress and recovery phases. It must be noted that the recovery is never complete. Existing literature [65] has noted that stressing a device even for 1% of the time followed by recovery phase for 99% of the time is still sufficient to slowly build up interface charge. However, the recovery phase is very important to be considered for correct estimation of NBTI effect. The

lifetime estimation of NBTI without considering the recovery phase can be an order of magnitude lower than the actual value [35]. To account for the relative time of stress and recovery phases, a common method is to deal in terms of signal probability (SP). Typically, SP is defined as the ratio of time a signal is at logic HIGH. However, since NBTI effect happens due to negative bias (i.e. input gate voltage at logic LOW), *we will denote the ratio of time a signal remains at logic LOW* to the total clock period as SP.

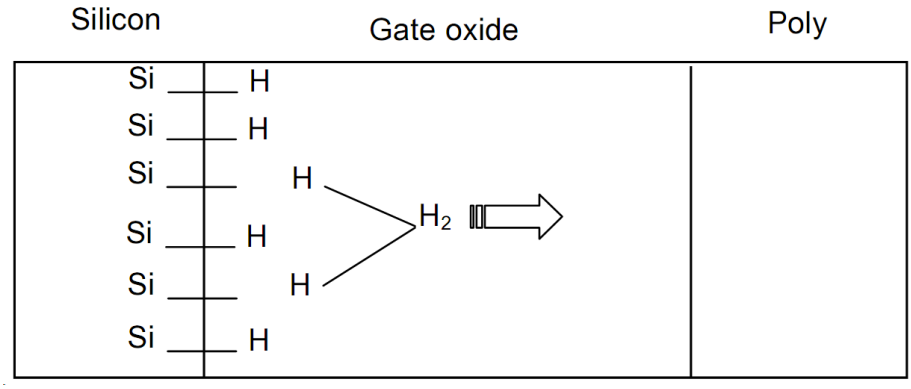

Figure 3.1: Illustration of NBTI phenomenon. Breaking of SiH bonds creates Si^{+} interstitial causing V_{th} increase. Picture taken from [16].

Clock is one of the most critical signals in the VLSI chip and special attention is given to generate reliable, low power clock trees while meeting stringent skew constraints. Clock skew is defined as the maximum difference in the arrival times of the clock signal at all those sequential elements (flops) which can interact with each other due to the presence of a path between them. A large value of clock skew implies that the clock signal reaches the flops at very different times. Clock skew has one-on-one impact on the maximum frequency

of operation of a chip, therefore decreasing it is a major design concern. The minimum period of the clock signal is given by the following formula:

$$ClockPeriod >= T_{cq} + T_{pd} + T_{skew} + T_{setup} \tag{3.1}$$

where T_{cq} is clock to Q delay of the flop, T_{pd} is propagation delay through combinational block, T_{skew} is clock skew and T_{setup} is the setup time of capturing flop. As can be seen from Equation 3.1, reducing clock skew directly reduces the minimum time period of operation. A high quality low skew clock tree often requires symmetric clock tree topology and advanced clock signal routing with the aim to equalize the arrival times of the clock signal at all the sequential elements. In recent times, due to several effects such as parametric variation, lack of high quality interconnect models and environment factors such as spatial temperature variation, maintaining low clock skew has become a challenge. The challenge is to come up with schemes such that all the path delays can be made as close to each other as possible thus maximizing frequency of operation.

All modern clock trees employ clock gating technique which selectively shuts down unused parts of the clock tree. Typically integrated clock gating (ICG) cells are inserted in the design which conceptually are composed of a latch followed by AND/OR gate. The presence of latch avoids glitches and premature ending of clock signal. When using AND or NAND gate as the output stage of ICG, the value coming out of latch should be the controlling value i.e. logic LOW. Similarly, when using OR or NOR gate at output stage

of ICG, the controlling value of logic HIGH needs to come out of latch. As the clock signal switches every cycle, the PMOS devices in clock buffer experiences alternate stress and recovery phases of equal duration. However, PMOS devices which are part of heavily gated clock buffer do not experience stress and recovery for equal durations. The degradation of V_{TH} in these clock buffers can go out-of-sync compared to the rest of the clock buffers. This non-uniformity in V_{TH} degradation can cause substantial increase in clock skew of a clock tree causing timing violations.

For example, consider the clock tree shown in Figure 3.2 where the shaded subtree is currently gated by active low GATE signal. As a result of this, though the PMOS device inside a non-gated clock buffer experiences alternate NBTI stress and recovery cycles, the PMOS inside a gated buffer undergoes constant stress. This may cause increase in skew of the whole clock tree. Also of importance is the workload dependent temporal variation in the temperature of the clock buffers. Due to strong dependence of NBTI effect on duty cycle and temperature, the study of its impact on skew degradation of clock tree is strongly mandated. This forms the motivation of our work.

The NBTI effect increases dramatically at higher operating temperature and supply voltage. In the whole manufacturing flow, burn-in testing is a unique stage when the VLSI product is simultaneously subjected to both these detrimental conditions. Thus, burn-in testing has potential to cause significant NBTI degradation (i.e. V_{TH} increase) of the devices even if the duration of burn-in testing is short (of the orders of tens of hours). We tackle the prob-

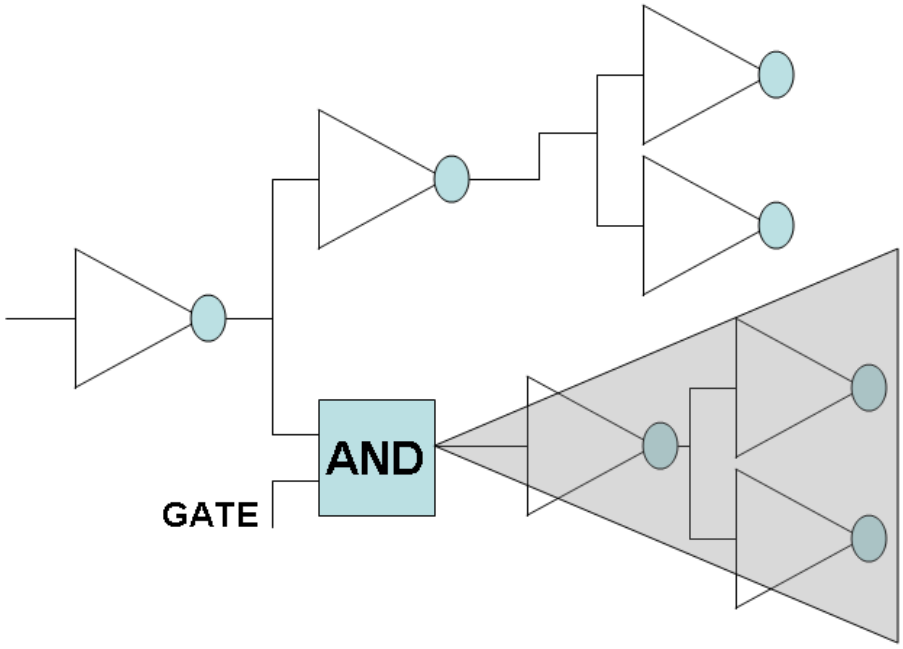

Figure 3.2: Example of Clock gating. Alternate PMOS in shaded sub-tree see constant stress for long time.

lem of quantifying and reducing the impact of burn-in testing on the NBTI degradation of a circuit. We identify that *static* burn-in testing in particular causes severe NBTI degradation of the chip since due to the static nature of the test (i.e. no switching), there are no recovery phases that can reduce the degradation. To the best of our knowledge, there is no prior work to reduce this detrimental effect. We propose finding a partially specified input vector to be used during burn-in testing with the aim of re-distributing the NBTI degradation based on timing criticality as well as NBTI sensitivity of different cells in the circuit.

3.2 Design-time Control of NBTI Aging of Clock Tree

In this section we will outline a design time approach to reduce the impact of NBTI aging on the skew degradation of clock tree. The aim of the proposed design technique is to reduce the skew of a clock tree arising due to the asymmetry in the V_{TH} degradation of the clock buffers. This asymmetry is due to difference in the signal probability (SP) at different parts of the clock tree due to clock gating. We propose to use *both* NAND and NOR gates[1](instead of just one of them) to implement clock gating. Using NAND (NOR) gate to shut down the clock allows freezing the clock tree at logic HIGH (LOW) thus decreasing (increasing) the signal probability for all clock buffers in the fanout cone. By intelligently choosing which gate to use for each clock gating element, the signal probability of the clock tree can be modulated to reduce the clock skew.

The input to our technique is a clock tree constructed using inverters or buffers. Using RTL simulation, clock gating opportunities at some of the clock inverters are identified. Normally, these inverters would be replaced by ICGs (integrated clock gates) with NAND gates at their output stage with the second input of NAND gate tied to active LOW clock gating enable signal. However, in our approach we would replace some of these ICGs with those which have NOR gate at their output stage. We next present an example to

[1]Using NAND or NOR gate is equivalent to using AND or OR gate with inverted signals. Since NAND/NOR gate are single stage like inverter, we prefer to use them. In clock trees containing buffers instead of inverters, our technique will choose between AND or OR gate.

demonstrate the possibility of skew reduction by selecting ICGs with NAND or NOR gates.

Motivating Example: Consider the clock tree shown in Figure 3.3 that drives four latches. The clock tree nodes (represented as inverters indexed by name under it) that have clock gating ability are circled and referred to as gated nodes. The nominal skew of the clock tree is zero due to symmetry. For the gated nodes, the probability of clock gating (G) of each node is also shown. A value of G=0 implies that this particular node is never gated, whereas G=1 means that clock is always gated. Assuming 50% duty cycle of clock at input (i.e. S_{in}=0.5), we computed the skew at clock tree leaves using HSpice after aging the circuit by 10 years. When all the gated nodes are implemented as NAND gated ICGs, the skew of the clock tree is 1.90 picoseconds (ps) whereas for a configuration of all-NOR gated ICGs, the clock skew is 1.36 ps. The best configuration is obtained when N2=NOR, N3=NOR, N5=NAND with a value of skew of only 0.16 ps, a reduction of almost 90%. This proves that simply choosing all gates as NAND or all gates as NOR is not the right choice.

3.2.1 NAND/NOR Aware SP Propagation

In this section we set up the ground rules for propagation of signal probability (SP) and delay when implementing clock gating through NAND or NOR gates. We would like to reiterate that we are defining SP of a signal as the ratio of time it is logic LOW, different from the conventional opposite meaning. Consider a clock tree inverter shown on left side in Figure 3.4. The

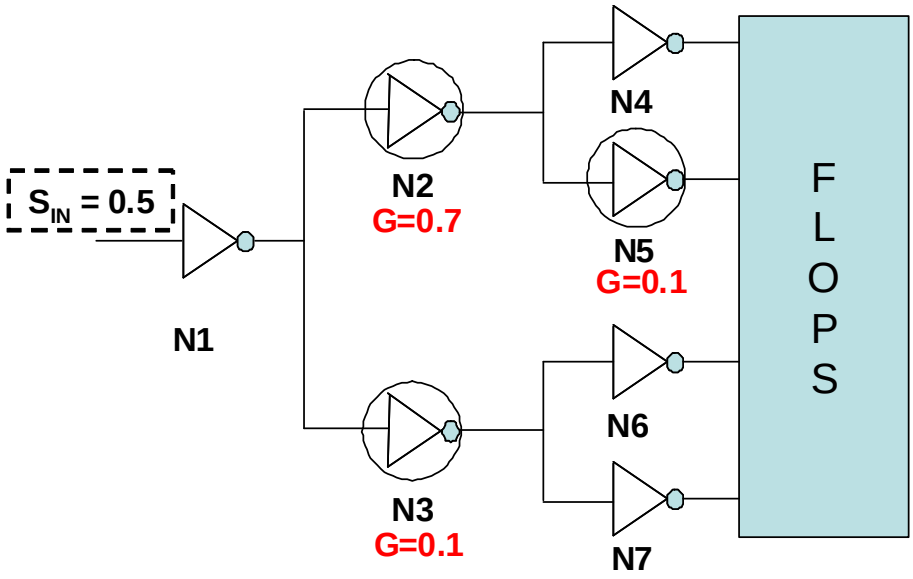

Figure 3.3: Example of a small clock tree. Nodes that allow clock gating are circled and their probability of clock gating indicated by symbol G. Input signal probability was set as 0.5 (i.e. nominal clock signal).

input SP of the inverter is S and the probability of clock gating is G. If this inverter is replaced by an NAND gated ICG, the SP of the clock gating signal would be G itself because logic LOW is the controlling value for NAND. In such a scenario, the output SP of the NAND gate is (1 - G)*(1 - S). On the other hand, if the inverter is replaced by a NOR gate, the SP of the clock gating signal would be (1 - G). This is because logic HIGH is the controlling value of NOR and gating probability of G means (1 - G) period of non-gating when the logic LOW is present. The output SP can then be obtained as 1 - S*(1-G). Let the binary variable X represent this choice between using NAND or NOR gate. X = 1 implies using NAND gate for clock gating and

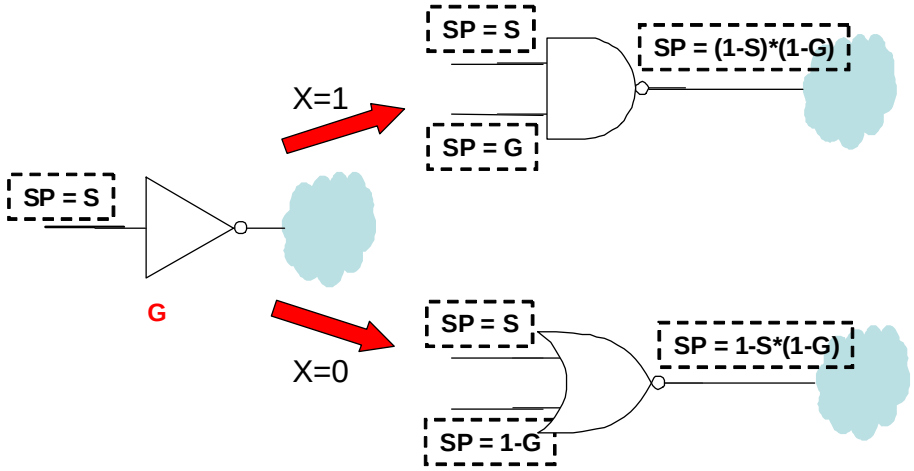

Figure 3.4: A clock inverter (left) with input signal probability of S and probability of clock gating G can be replaced by either a NAND gate (right top) or a NOR gate (right bottom) with the indicated signal probability (SP) at their inputs. The output SP is also shown for both the cases.

X = 0 implies choosing NOR gate. For the regular inverters in the clock tree (i.e. those that do not have clock gating ability), the output SP is trivially equal to (1-S). Let us assume the delay of an INV, NAND and NOR gate is D_{INV}(S,G), D_{NAND}(S,G), and D_{NOR}(S,G) respectively, which are functions of the switching probability (S) and Clock Gating probability (G). If X = 1, the delay of the cell is D_{NAND}(S, G), else it is D_{NOR}(S, G). Delay of a clock inverter which does not clock gating capability is simply D_{INV}(S, G). Table 3.1 summarizes these observations which are used for propagating the symbolic signal probability and delay values through the clock tree from the root to the clock leaves.

The information from Table 3.1 can be combined to get the following expression for the output SP and delay through a clock gating enabled gate in

Table 3.1: Formula for output SP and delay of different gates. Binary variable X = 1 if NAND is chosen, 0 if NOR is chosen.

Choice	Variable X	Output SP	Delay
NAND	1	(1-G)*(1-S)	D_{NAND}(S, G)
NOR	0	1-S*(1-G)	D_{NOR}(S, G)
INV	-	1-S	D_{INV}(S, G)

terms of the binary variable X.

$$SP_{out} = 1 + S * G - S - X * G \tag{3.2}$$

$$D = X * D_{NAND}(S, G) + \overline{X} * D_{NOR}(S, G) \tag{3.3}$$

We prove an important property of the delay expression of any gate which will help making the ILP formulation tractable later in this chapter.

Lemma 1. Signal probability of any gate is at most a linear function of X.

Proof. Consider Eqn 3.2. If input signal probability, S, is linear or constant in X then output signal probability SP_{out} is also linear since G is a constant. As SP_{out} becomes the input signal probability for fanout gate, the linearity property remains recursively true. As the base case, signal probability of clock tree root is a constant number. □

Lemma 2. If cell delay is linearly dependent on input signal probability, delay expression of any gate is at most a quadratic function of X.

Proof. From above lemma, signal probability is linear function of X. From Equation 3.3, delay expression is linear combination of two expressions where

X is multiplied by the delay of the NAND or NOR cell. As long as the delay of each cell is linear function of input signal probability, the delay expression is at most a quadratic function of X. □

Using the above expressions as well as that for inverter from Table 3.1, we can start at the root of the clock tree and recursively compute the symbolic signal probability and the delay from the root of the clock to each leaf level sink. An example, of this is as follows:

Example 2. Consider the toy clock tree shown in Figure 3.5 where each clock tree node that implements clock gating is circled. Let the binary variable X2 and X4 represent the choice of NAND/NOR gate at the nodes N2 and N4. The probability of clock gating (G) is written next to the corresponding nodes. The propagated signal probability of the path from clock root to the clock tree leaves are computed for each net based on G and the binary variables X2 and X4 and noted along the net. Ignoring dependence of delay of a NOR gate on G[2], the delay at upper leaf can be written as [$D_{INV}(0.5)$] + [$X2$ * $D_{NAND}(0.5)$ + $\overline{X2}$ * $D_{NOR}(0.5)$] + [$D_{INV}(0.75 - X2*0.5)$]. Similarly, the delay at the lower leaf can be written as [$D_{INV}(0.5)$] + [$X2$ * $D_{NAND}(0.5)$ + $\overline{X2}$ * $D_{NOR}(0.5)$] + [$X4$ * $D_{NAND}(0.75 - X2*0.5)$ + $\overline{X4}$ * $D_{NOR}(0.75 - X2*0.5)$].

From the above example, we note that the choice of using NAND and NOR gate (i.e. variable X2 and X4) not only modifies the delay function of the clock path but also modulates the signal probability (SP) at the fanout cone

[2]This dependence is characterized later in Section 3.2.2

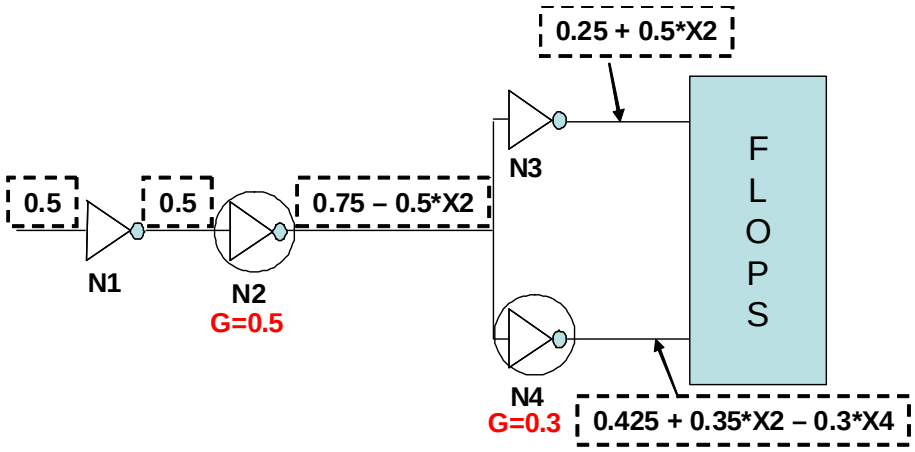

Figure 3.5: Example showing the propagated value of SP (in dashed boxes) as a function of X2 & X4 indicating the choice between NAND and NOR gate at nodes N2 and N4 respectively. Clock gating probability used for calculation is represented as symbol G.

affecting the delay of output gates. Therefore, properly making this choice can help in reducing clock skew.

3.2.2 SP & GP Aware Delay Model

We sized the NAND and NOR gates to match their rise and fall delay to those of an INV (inverter). In this way, replacement of the INV by any other gate will not change the nominal clock skew. The ratios of PMOS to NMOS width for INV, NAND, and NOR gate in our library that achieved this iso-delay setting are 2.2, 1.36, and 4.46 respectively. The delay computed through HSpice has a nominal value of 22.69 ps for fanout-4 load at 50^{o}C. Next step is to characterize the delay of these cells as function of clock signal probability (SP) and gating probability (GP). Since NBTI does not impact

the output load capacitance of the gate in any way, the load dependent delay is ignored for degradation analysis. Our aim for delay characterization is to extract simple high fidelity approximations to guide the optimization engine in right direction. Therefore, we extensively use linear approximation of near-linear curves.

To consider the impact of SP on delay, we first need to relate SP to V_{TH}. We computed the V_{TH} degradation as a function of SP using the s_k model of [64] extensively employed in other works such as [29][65] etc. Using the obtained V_{TH} values, we performed spice simulation to obtain the rise and fall delay of the NAND, NOR and inverter gate. Since NBTI impacts only PMOS devices, the fall delay of the gates was observed to be nearly constant for all SP. However, the rise delay of these gates varies by as much as 10% when the SP increases from 0 to 100%. The dashed curves in Figure 3.6 shows the rise delay of INV, NAND and NOR gate as a function of SP.

There is a large increase in delay degradation near very low value of signal probability of approximately 5%. However, the curve flattens out for larger values of SP. This observation is consistent with those obtained by other authors such as in [65]. Similar to [65], to model this behavior, we performed piecewise linear fit for the case of SP $\leq$ 5% and for SP $>$ 5% obtained through Gnuplot tool with R^2 coefficient of fit as 0.88. These linear fits are as follows:

$$D_{INV}(SP) = \begin{cases} (0.4428\text{*SP} + 22.69)\,\text{ps} & : SP \leq 0.05 \\ (0.0417\text{*SP} + 24.79)\,\text{ps} & : SP > 0.05 \end{cases}$$

$$D_{NAND}(SP) = \begin{cases} (0.4213\text{*SP} + 22.69)\,\text{ps} & : SP \leq 0.05 \\ (0.0410\text{*SP} + 24.69)\,\text{ps} & : SP > 0.05 \end{cases}$$

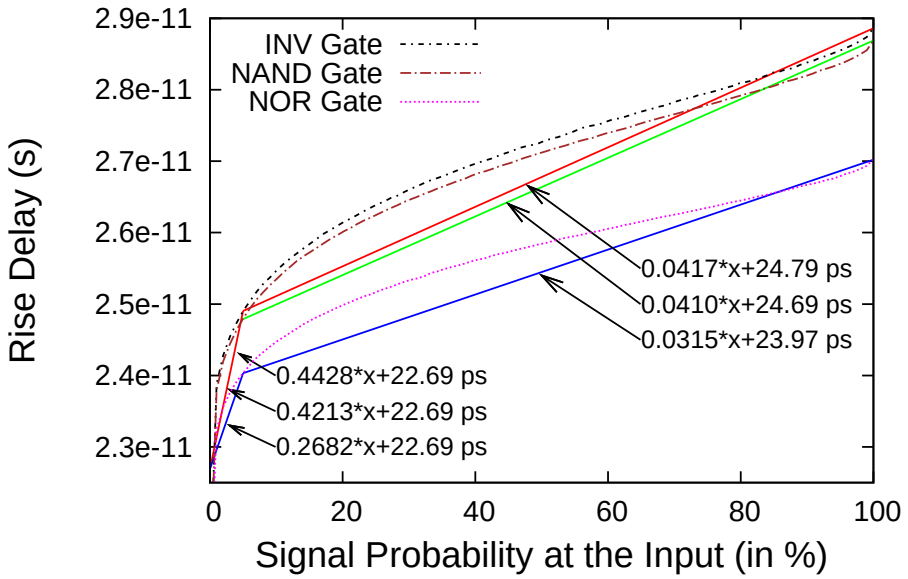

Figure 3.6: Rise delay of INV, NAND and NOR gate as a function of input signal probability (SP). Initial sharp increase is observed. Change in slope near 5% SP motivated the piecewise linear delay model.

$$D_{NOR}(SP) = \begin{cases} (0.2682\text{*SP} + 22.69)\,\text{ps} & : SP \leq 0.05 \\ (0.0315\text{*SP} + 23.97)\,\text{ps} & : SP > 0.05 \end{cases}$$

From the previous discussion it is clear that clock signal probability has direct impact on the delay of the fanout gate. Next we consider the impact of gating probability (GP) of a NAND/NOR gate on its own delay. Both NAND and NOR gates have two PMOS transistors driven by two separate pins. One of the input pin is driven by the clock signal from previous stage of clock tree with signal probability of SP and the other pin is driven by gating enable signal latched in the ICG (integrated clock gate) with signal probability equal to GP. In the case of NAND gate, due to parallel paths to V_{DD} through

the two PMOS devices, even if the PMOS connected to gating enable signal degrades, the net impact on the rise time is negligible. On the other hand, in the case of NOR gate, different values of gating enable probability lead to different V_{TH} degradation of the PMOS driven by gating enable signal. This directly affects the pull-up capability of the NOR gate due to the inherent PMOS stack in it. In short, for NOR gate we must consider the impact of degradation of both PMOS transistors. This was first pointed out by [65]. To capture this effect, we simulated the rise delay of NOR gate as a function of the V_{TH} degradation of PMOS driven by input clock for different gating enable probabilities driving the second PMOS input. Figure 3.7 shows the rise delay of NOR gate as a function of the gating probability at it on the x-axis (i.e. how frequently it is gated), for different signal probability of the clock pin. From this figure, it is immediately visible that gating probability can play significant role in determining the rise delay of the NOR gate due to the stacking effect. Fortunately, in the most relevant range of signal probabilities the delay dependence can be very closely approximated by a linear dependence.

Clearly, the higher is the gating probability at the gating enable signal (i.e. of clock being gated), the higher is the proportion of the time logic HIGH (controlling value for NOR gate) is fed to the NOR gate which translates into *lower* NBTI degradation. The rise delay of the NOR was observed to decrease approximately 8% in a near-linear manner when the clock gating probability varies from 0% to 100%. Hence, we incorporated this gating probability

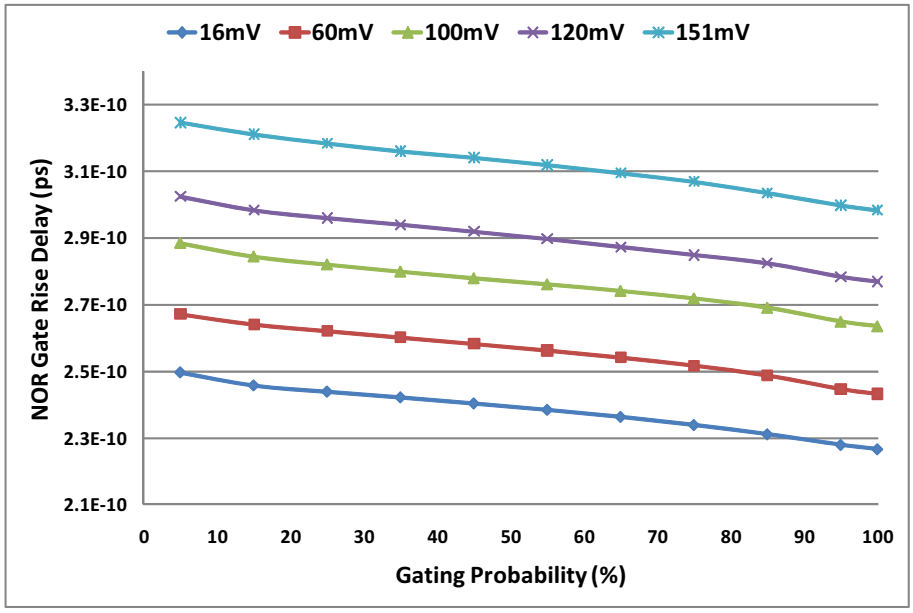

Figure 3.7: Rise delay of NOR gate as a function of clock gating probability for various NBTI degraded V_{TH} values at the clock input pin.

dependence by linearly scaling the NOR delay as follows [3]:

$$D_{NOR}(SP, GP) = D_{NOR}(SP) * (1 - 0.08 * GP) \tag{3.4}$$

Temperature has two different mechanisms of impacting gate delay: a) drive capability of the device and the threshold voltage changes with temperature, and b) temperature dependent NBTI further changes the threshold voltage. To capture the second mechanism, we make use of a relationship derived in [23] which provides a very tight upper bound for estimation of threshold voltage degradation of a PMOS device over a long period of time due to NBTI

[3] As explained earlier, NAND gate and INV gate do not have any dependence on GP.

degradation. NBTI induced threshold voltage degradation, ΔV_{th}, is given by

$$\Delta V_{TH,t} \propto \left((V_{GS} - V_{th})^2 \, e^{\frac{-2E_a}{kT}} \right)^{\frac{1}{6}} \tag{3.5}$$

where V_{gs}, V_{th} are gate-to-source voltage and threshold voltage, E_a is the activation energy ($=0.49eV$ for H_2 diffusion), k is Boltzmann constant with a value of $8.6174X10^{-5}$eVK^{-1}and T is the absolute temperature in Kelvin. Using this, ΔV_{th} at temperature T can be written in terms of ΔV_{th} at temperature T_{ref} as

Using the expression for dependence on signal probability (SP) and gating probability (GP), now we can analytically write the delay of each of the three cells for any combination of these variables. These expressions can be used for optimizing clock skew of large scale circuits using integer programming formulation described next.

3.2.3 Skew Reduction Formulation

Using the models developed in the previous section, we will present our optimization program formulation for skew reduction of a clock tree in presence of NBTI degradation after 10 years of aging. Let the set of sinks in the clock tree be given as S. For the i-th sink s_i, using the piecewise linear delay model developed in Section 3.2.2, we can obtain the formula for arrival time of the clock signal. Obviously, the arrival time is a function of the signal probability of the clock interconnects and gating probability of clock buffers connecting sink s_i to the clock signal root. This can be represented as $AT_i(X_i, SP_i)$, where

X_i and SP_i capture the binary variables for the choice of NAND/NOR clock gating and signal probabilities along the path.

There are two interesting problems that can be formulated. The first is an *optimization* problem to identify the optimal choice of NAND/NOR gate configuration to minimize the skew of the clock tree. This case is of special importance for high performance designs or when the clock tree structure is already fixed but timing closure is difficult to achieve. The second problem is a satisfiability (SAT) problem which, given a clock tree structure, decides whether there is any configuration of NAND/NOR assignment that meets a particular skew constraint after circuit aging. We will focus on the *optimization* problem due to its more practical use. Consider the following formulation:

$$\begin{aligned} Minimize &: (MAX - MIN) \\ Subject\ To &: \\ AT_i(\{X_i\}, \{SP_i\}) &\leq MAX \quad \forall i \in S \\ AT_i(\{X_i\}, \{SP_i\}) &\geq MIN \quad \forall i \in S \\ X_i &\in \{0, 1\} \quad \forall i \in S \\ MAX &\geq 0 \\ MIN &\geq 0 \end{aligned} \tag{3.6}$$

In this, `MAX` and `MIN` are dummy variables which represent the largest and the smallest arrival time of the clock signal among all sinks as indicated by the first two constraints over all sinks. All the X_i variables are constrained

to be binary. By minimizing the objective (`MAX - MIN`), we are effectively minimizing the clock skew of the whole clock tree. For a clock tree with n sinks, the number of constraints are clearly O(n). Assuming balanced tree structure, there are log(n) levels thus each of the $\mathrm{AT}_i(\mathrm{X}_i, \mathrm{SP}_i)$ has at most O(log(n)) binary variables.

The expression of arrival time contains multiplication of binary variables which can cause solvers to fail. In Section 3.2.1 we proved that the delay expression can have multiplication of at most two binary variables. To decompose such expressions, we use the following transformation. Let X_A and X_B be the two binary variables whose multiplication appears in arrival time expression. We introduce a new binary variable X_{AB} such that

$$\begin{array}{lcl} X_A + X_B & \leq & 1 + X_{AB} \\ (1 - X_A) + (1 - X_B) & \leq & 2 - 2 \times X_{AB} \end{array}$$

By replacing $\mathrm{X}_A \times \mathrm{X}_B$ by X_{AB}, and adding the above constraints to the ILP, the new problem is equivalent but without any multiplication of binary variables.

Consider a design with 100K cells out of which 10%, i.e. 10K of these cells may be flops. Assuming each leaf level clock buffer can drive 10 flops, the clock tree will have 1K leaf level clock buffers or approximately 2K total clock buffers in the clock tree. Even if 10% of these clock buffers are gating enabled, we have at most 200 binary variables. Therefore, the above formulation can be solved readily by solvers, branch and bound techniques or simulated annealing methods to get the optimal solution for skew minimization.

3.2.4 Experimental Setup & Results

For all our transistor level simulations, we used the post extraction spice models from the open-source 45nm Nangate library [11]. Simulations were run using Synopsys HSPICE Version A-2008.03-SP1. A C++ program was written to symbolically propagate the signal probability, to compute the symbolic delay equation of each clock sink and to write the optimization problem. We used tool CPLEX to solve our optimization problem. For managing the long symbolic expressions, we used the symbolic expression simplifier built in the tool Mathematica. All the above steps were performed on dual core 2.67 GHz workstation running Linux operating system.

For benchmarks, we generated several instances of clock trees with varying levels (depth of clock tree) and the fanout number of each clock buffer. Approximately 2% of the clock buffers were randomly picked to be gating enabled. The gating ratio of these buffers was chosen randomly between 20% and 70%. The reason we do not allow clock gating ratio of less than 20% is because clock gating does not come for free. There can be appreciable power dissipation in the logic gates that generate a signal whether a certain logic block must be gated in next clock cycle or not. The power dissipation of such auxiliary gates must be weighed in against the possible savings in power by clock gating of that module. In other words, we assume that if a module is going to be turned off less than 20% of the times, the cost of clock gating signal generation may be more than power savings expected. The input signal probability to the clock tree root was assumed to be 50%. Because of perfect

symmetry of output load and matching of delay of NAND/NOR gate to that of inverter, the initial clock skew for all benchmarks is 0ps. Table 3.2 shows the characteristics of the benchmarks used in this work. In the table, the second column onwards contain the depth of the clock tree, the number of fanout nodes of each clock buffer, the total number of buffers and flops at the sinks of the clock tree. The last column shows the number of buffers that have clock gating capability. Each one of such buffers are associated with one binary decision variable for choosing NAND or NOR implementation.

Table 3.2: Clock tree benchmarks used in this work. Depth of the tree, the fanout of each clock inverter, the number of buffers, sinks (flops) and clock gating enabled inverters shown in consecutive columns.

Name	Depth	Fanout	# Buf^{+}	# Sinks*	# Gated
A	7	4	21845	87380	331
B	8	3	9841	8748	144
C	9	3	29524	26244	426
D	8	4	87381	348520	1251
E	9	3	29524	26244	430
F	8	3	9841	8748	138
G	8	4	87381	348520	1267
H	7	4	21845	87380	326

$^{+}$Number of buffers include all levels, not just leaves.
*Number of sink assuming fanout of 4 at leaves.

Table 3.3 contains our results. Column 2 shows the CPU time to run ILP solver on our optimization program. The next two columns show the fraction (in %) of the clock gating buffers converted to NAND and NOR gates respectively in our solution. The skew reported by our optimization flow is shown in column 6. Next, we compare our solution to the three strategies:

choosing all NAND gates (symbol $\forall$ NAND), choosing NOR gates (symbol $\forall$ NOR), and running 10 random assignments of NAND and NOR gates and picking the best among these. For each one of these strategies has two columns in the table. The first one reports the skew value obtained by that strategy and the second column contains the % penalty in skew that this strategy has over our optimal solution.

From the skew numbers in Table 3.3, we observe that our proposed method gives very good results. As compared to our optimal clock gating solution, the clock skew of designs implemented using all-NAND, all-NOR, and random choices are on an average 56%, 219% and 133% higher. This proves that the use of our method can significantly tighten the skew budget helping high performance designs. In some of the benchmarks (see benchmark D in Table 3.3), the optimal solution was only 37% better than the trivial solution of using all-NAND gates. However, on other occasions, our solution was up to 74% better. In general, the skew when using only NAND gates was lesser than using only NOR gates. We believe this is due to the different delay dependence curves of NOR gate in Figure 3.6 compared to the seemingly similar curves for INV and NAND gates. However, it does not mean that the optimal solution has most of the clock gated buffers chosen to be NAND gates. From the runtime perspective, we note that the maximum CPU runtime for solving these testcases is less than 2 seconds. Benchmark G which has the largest number of gated clock buffers is solved in less than half a second.

Validation: In Section 3.2.2, we extensively used linear approxima-

tions for developing easy model for optimization program generation. In this process, we would have lost some accuracy. However, as long as the optimization program is guided in right direction, a good solution will be achieved. To check how accurate our approximations were, we did the following experiment. For all benchmarks, we computed the clock skew directly using spice simulation of the fastest and slowest clock paths for each of the three configurations: all NAND, all NOR and the optimal configuration obtained by us. Though the exact skew numbers were different from the numbers reported in Table 3.3, the penalty of using the configuration of all-NAND and all-NOR gates matched with our results. For example, in the case of benchmark D, the skew penalty reported by our model is 56% and 219% for all-NAND and all-NOR respectively, while the HSpice returned numbers are 58% and 216% respectively. This proves that our linearized model has good fidelity and can be used for skew optimization.

3.2.5 Impact of Variability

High quality variation tolerant clock tree design relies on making the clock tree as symmetric as possible. This is achieved by having the same *signature* of each clock source to sink (i.e. latch) path. This signature is comprised of the interconnect length, number of clock buffers inserted, and the sizing of the clock buffers used in each source to sink path. In a clock tree without clock gating, it is relatively easier to obtain such symmetry though at the cost of power or chip area. In the case of traditional clock gating,

the clock trees is exclusively made up of clock buffers and integrated clock gating elements with NAND gate at their output. Even for this clock gating scheme, there is inherently a source of asymmetry because not all paths from clock source to sink have clock gating enabled on them. The clock tree with traditional clock gating scheme comprises of INV and NAND gates and the delay response of these two types of gates to various variability sources could be different. This asymmetry can lead to increase in clock skew in the presence of variations.

One concern with our proposed approach is that instead of *two* types of gates that are used in traditional clock gating methodology (i.e. INV and NAND), the proposed technique uses *three* types of gates by adding NOR gate in the clock tree. This can further increase the asymmetry between the different paths in the clock tree leading to higher skew. The results from Section 3.2.4 shows that our proposed technique can significantly reduce clock skew due to NBTI degradation of the various gates, however that analysis is done without considering any device or operation condition variations.

To fully understand how the proposed technique performs in presence of variation, we performed the following experiments. We identified a few sources of variation and subjected the clock trees to these variation conditions one at a time. The skew of the two clock trees a) traditional clock gating (with INV and NAND gates) and b) proposed approach (with INV, NAND, and NOR gates) was computed under these operating conditions. In all cases, it was assumed that the clock skew at the beginning of the lifetime of the

clock tree (T=0) *under nominal condition* is zero which is practical because typically clock buffer library is rich and the correct gate sizes for the gates can be picked to satisfy this condition. Surely, under variation even the starting skew is non zero. The value of clock skew under variation computed in this section is at end of life (i.e. T=10 years) of the VLSI product.

3.2.6 Temperature Variation

In this experiment, we changed the spatially uniform operating temperature of the clock tree and computed the clock skew. Though it is possible to also vary the temperature spatially, however, as the different clock gates (INV, NAND, NOR) can be spread randomly across the chip, the difference in their delay variation due to different sensitivity to temperature variation can possibly average out. In that sense, spatially uniform temperature variation gives a deterministic way to capture the impact of variation. Figure 3.8 shows the variation in the delay of the three types of gates as a function of operating temperature. It is evident from the figure that the behaviour of the NAND gate remains similar to that of INV gate whereas the delay of NOR gate diverges slightly more. Since skew is computed by difference in the delay of different clock paths, even small difference in the absolute delay values of the gates can translate into significantly higher skew. We computed the clock skew for our benchmark circuits for operating temperatures of 60^oC, 80^oC, and 100^oC respectively. Figure 3.9 shows the clock skew by using the traditional clock gating that uses INV and NAND gates vs our proposed technique that

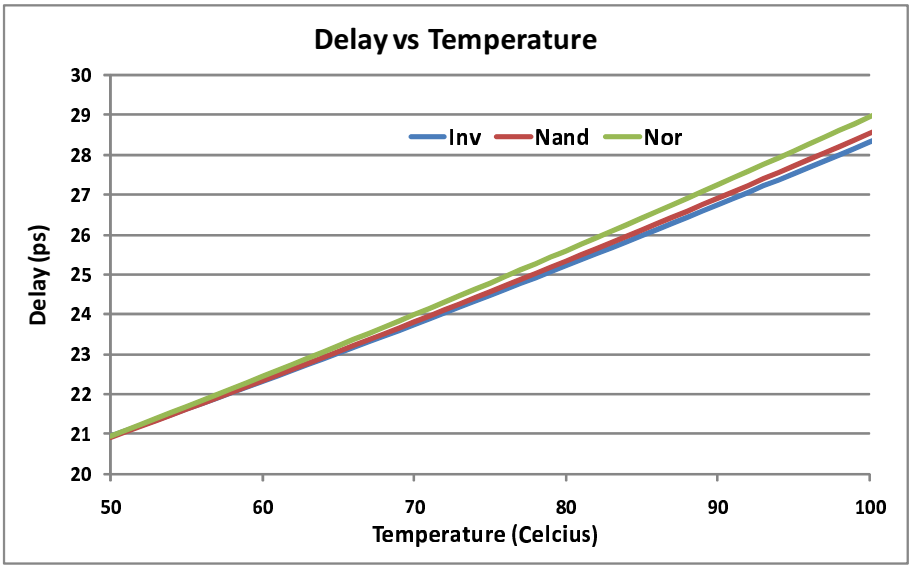

Figure 3.8: Delay of INV, NAND, and NOR gate as a function of operating temperature.

uses INV, NAND and NOR gates under temperature variation. To preserve image quality, only the data for benchmarks A, C, and E are shown though the data from other benchmarks follows similar trends. From Figure 3.9, we observe that with variation in operating temperature, in general the saving in clock skew achieved by our approach reduces as compared to the savings at nominal operating condition. Due to different temperature sensitivity of the constituent gates (as seen in Figure 3.8) the fastest and slowest path from root to the clock tree sinks can change at different temperatures. This can lead to some outliers such as benchmark "C" in Figure 3.9 for which the skew reduction actually improves at lower temperature. This is because while the traditional gating suffers from different delay variation of NAND and INV gates, clock tree built with our approach also suffers from delay variation of

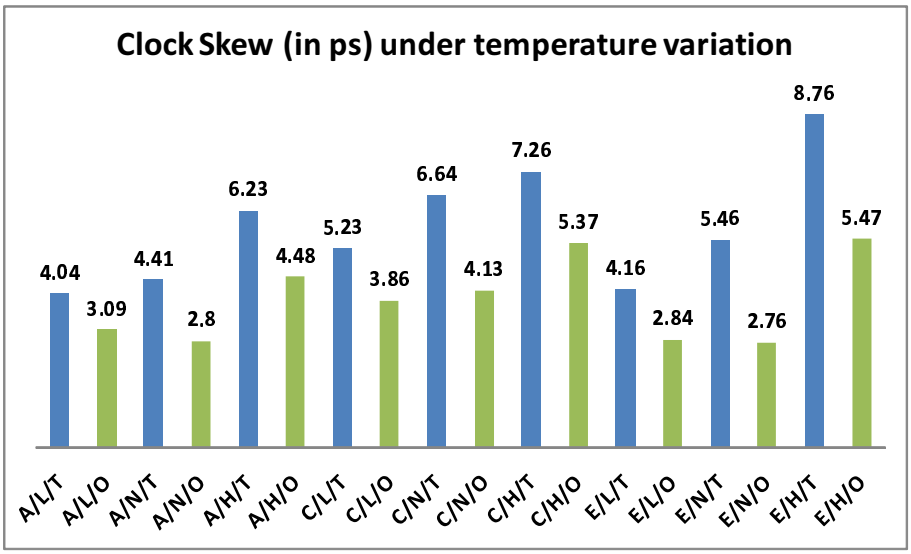

Figure 3.9: Clock skew for different benchmarks under temperature variation for benchmarks A, C and E. X-axis identifies three properties separated by '/'. a) the benchmark name(A, C, E etc), b) temperature type (L: Low, N: Nominal, H: High), and c) clock tree type (T: Traditional, O: Ours). All blue bars are for traditional clock gating, green bars are for our proposed scheme.

the NOR gates. Over these three benchmarks, the clock skew saving reduces from 41% at nominal condition to 27% at low temperature, and 30% at high temperature operating conditions.

3.2.7 Supply Voltage Variation

In this experiment, we changed the operating voltages of all the gates in the clock tree by a spatially uniform value. Figure 3.10 shows the variation in the delay of the three types of gates as a function of supply voltage. From Figure 3.10, it is evident that the delay of the NOR gate is most divergent with variation in supply voltage. Note that the all the three gates are made to

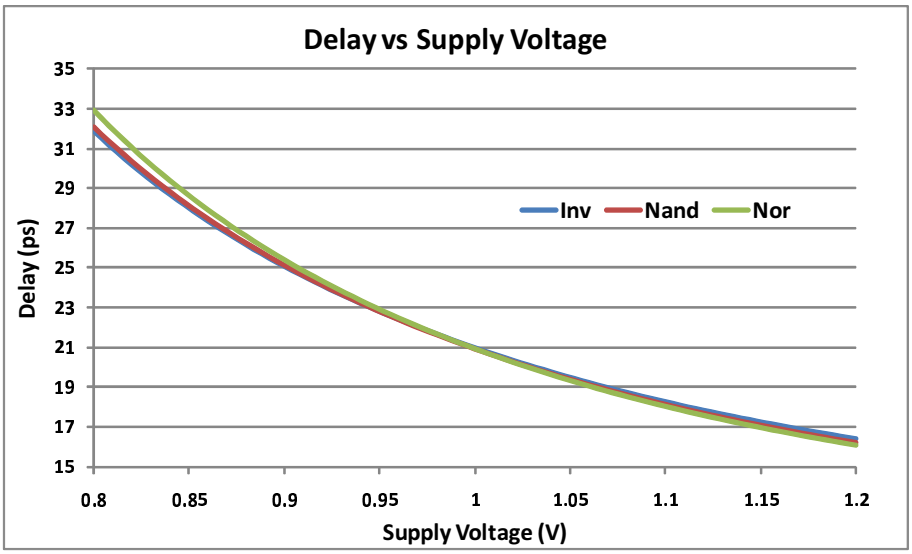

Figure 3.10: Delay of INV, NAND, and NOR gate as a function of operating supply voltage.

be iso-delay at nominal operating voltage of 1V. At operating voltages lesser than the nominal voltages, the delay of the NOR gate is the largest among the three gates. However, at operating voltages higher than the nominal voltages, the delay of the NOR gate is least among the three gates. We also notice that the behavior of the INV and NAND gate is relatively similar. Figure 3.11 shows the clock skew by using the traditional clock gating that uses INV and NAND gates vs our proposed technique that uses INV, NAND and NOR gates under operating supply voltage of 0.9V, 1.0V, and 1.1V. From Figure 3.11, we observe similar behaviour as observed for temperature variation. The saving in clock skew achieved by our approach reduces as compared to the savings at nominal operating condition under supply voltage variation. Over these three benchmarks, the clock skew saving reduces from 41% at nominal condi-

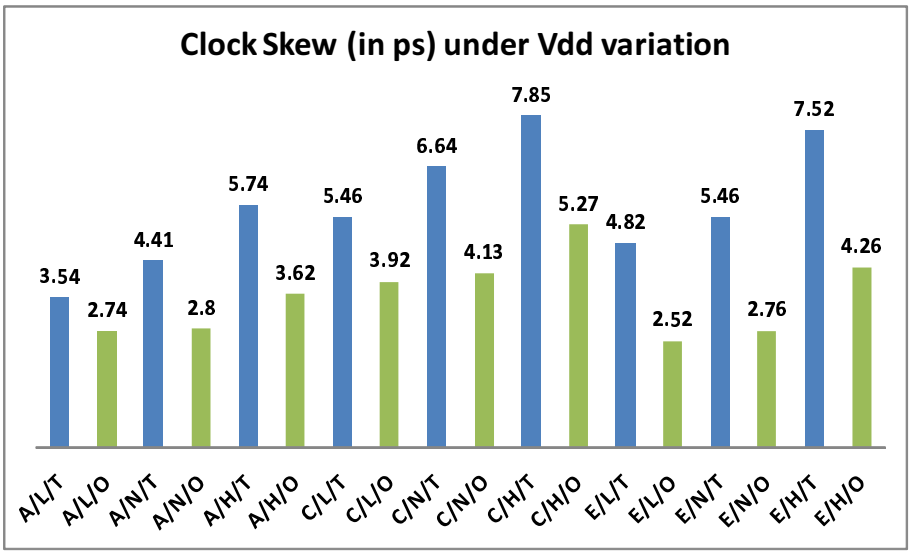

Figure 3.11: Clock skew for different benchmarks under voltage variation for benchmarks A, C and E. X-axis identifies three properties separated by '/'. a) the benchmark name(A, C, E etc), b) supply voltage type (L: Low, N: Nominal, H: High), and c) clock tree type (T: Traditional, O: Ours). All blue bars are for traditional clock gating, green bars are for our proposed scheme.

tion to 32% at low supply voltage, and 37% at high supply voltage operating conditions.

3.2.8 Discussions

In this section, we have proposed a static method for controlling NBTI degraded clock skew due to clock gating. Our method relies on design time intelligent choice of determining which clock buffer will freeze the clock at logic 0 (using NOR gate) or logic 1 (using NAND gate) during clock gating. This choice provides us two degrees of freedom: firstly, the choice of the gate changes the delay function of that clock branch and secondly, it modulates the

signal probability of the fanout cone, affecting the delay of gates downstream. We derived high fidelity piecewise linear models and corrective terms for computing the impact of signal probability at the input and the degradation of the clock gating element itself. The skew minimization problem was formulated as an integer linear program (ILP) and solved using commercial solvers. By exploiting our technique, we are able to reduce the NBTI induced clock skew by up to 74% compared to all-NAND implementation and 300% compared to all-NOR implementation. The impact of supply voltage and temperature variation on the proposed clock tree methodology was explored and we observed increased sensitivity to variation using the new approach due to presence of a different type (NOR) gate compared to the traditional clock gating structure.

3.3 Run-time Control of NBTI Aging of Clock Tree

To compute the effective temperature, T^{eff}, of a block for the purpose of NBTI degradation estimation, we need to capture the following two pieces of information: a) temporal distribution of temperature of the block during its operation, and b) cumulative impact of temperature dependent NBTI on this distribution. In this section we will demonstrate our technique to compute these two pieces of information and derive T^{eff} from it.

3.3.1 Capturing Temperature Distribution

For capturing the temporal variation in temperature of a circuit element due to different inputs to the circuit, we performed the following experiment:

Each primary input of our benchmark circuit was assigned a random switching activity and the circuit was assumed to run for 10 million cycles for this set of input switching activities. At the end of 10 million cycles, another set of random input switching activity was generated and the simulation was repeated. This process was performed 10000 times which correspond to total of 100 billion cycles. The power consumption of each cell was generated using program *Design Compiler* at each such interval of 10 million cycles. These power numbers were then coupled with the placement information of the chip to generate a sequence of *power dissipation maps* of the whole chip after dividing it into several grids (100 grids in our implementation). Thermal simulator *HotSpot* [53] was then used to convert these power map into a sequence of temperature maps.

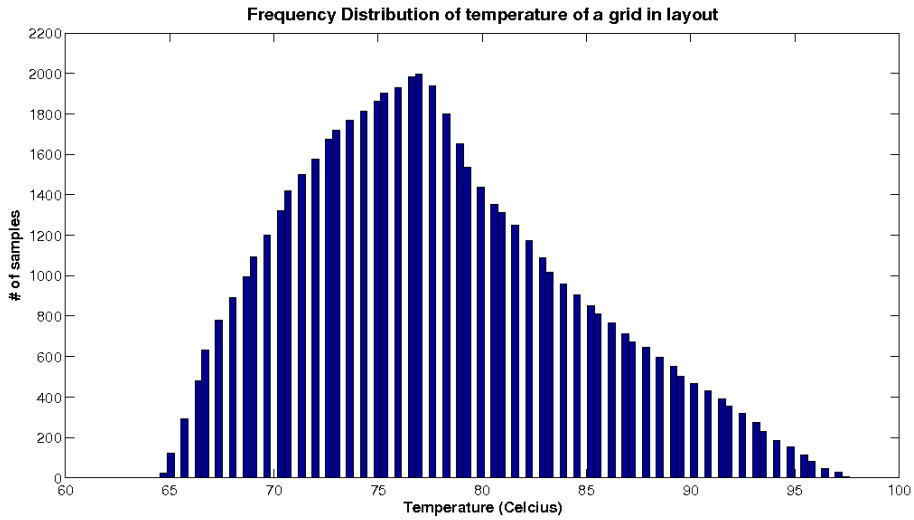

Figure 3.12: Frequency distribution of temperature of a grid. Sampling frequency = 1 million cycles.

On plotting the obtained temperature values for an arbitrary grid, we observed that the spread of temperature is near-gaussian with the value of mean and standard deviation varying for different grids[4]. For example, for the benchmark *des*, the spread of temperature of an arbitrary block is shown in Figure 3.12. In rest of the text, we will assume that the temperature of a cell i is gaussian with mean and standard deviation of μ_i and σ_i respectively. Thus, the probability distribution of temperature of a cell i, P(T_i) can be written as:

$$P\left(T_i\right) \approx \frac{1}{\sqrt{2\pi}\sigma_i} \times e^{\frac{(x-\mu_i)}{2\sigma_i^2}} \tag{3.7}$$

3.3.2 Temperature Dependent NBTI Impact

The complete temperature dependence of NBTI can be found by re-interpreting Eqn 3.5 as follows:

$$NBTI(T) = K \times \left(\left(V_{gs} - V_{th}\right)^2 \times e^{\frac{-2E_a}{KT}}\right)^n \tag{3.8}$$

where K is a constant, $E_a = 0.49$eV, n = 0.16, $V_{gs} = 1$V, K = 8.6174×10^{-7}eV/K. In our 45nm library [11], $V_{th} = -0.22$. Increase in temperature affects NBTI in two ways: a) following the Arrhenius dependency, the exponential term increase significantly, and b) V_{th} reduction with temperature causes further NBTI aggravation. As the cell library we used did not specify spice parameter TCV (temperature dependence of threshold voltage), we used its value as -1mV/^{o}C based on PTM and HSpice manuals [89]. Due to

[4]One may also fit the profile to a triangle. However, we observed that as the simulation points increase, the triangular profile developed into gaussian profile

exponential dependence of NBTI on the temperature of the circuit, one may be tempted to assume that negligible NBTI degradation happens at non-peak temperatures. However, in a typical operating temperature range of 300K-390K[5] of a cell([60]), the degradation at 300K is as much as 25% of that at 390K and thus not negligible. From this, we deduce that while calculating the NBTI degradation of a block, just considering the peak temperature can induce severe inaccuracy in lifetime prediction and the whole temperature range needs to be accounted for.

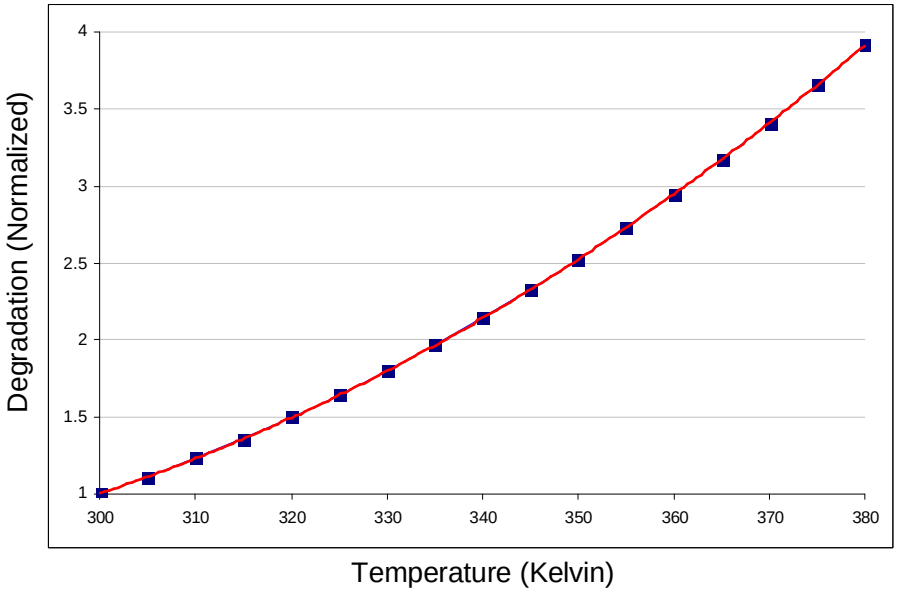

Figure 3.13: NBTI degradation normalized w.r.t. degradation at 300K (blue dots) and the quadratic fit (red line)

In the range of temperatures typically associated with VLSI chip opera-

[5]Much higher temperature variation of a block is observed when it is completely turned off using power-gating. However, during power-gating, there will be no NBTI impact.

tion, the dependence of NBTI on temperature using Eqn 3.8 can be very closely fitted by a quadratic curve. Figure 3.13 shows the original (using Eqn 3.8 and associated parameters), and the fitted quadratic curve of NBTI degradation of a cell in the range of 300K-380K normalized w.r.t. the degradation at 300K. At temperature T_i, the fit can be described as:

$$NBTI(T_i) \approx a \times T_i^2 + b \times T_i + c \tag{3.9}$$

where a=1.98×10^{-4}, b=-9.83×10^{-2}, and c=12.67. The R-square goodness of the above fit is 0.9999.

3.3.3 $\mathbf{T^{eff}}$ Computation

Using Eqn 3.7 and Eqn 3.9, the NBTI degradation of cell i is given by the expected value of product of temperature distribution curve and NBTI degradation at that temperature.

$$Degr(i) = E\left(NBTI(T_i) \times P\left(T_i\right)\right) \tag{3.10}$$

At very low temperatures, both $P\left(T_i\right)$ and $NBTI(T_i)$ are nearly zero. At very high temperatures, though $NBTI(T_i)$ is increasing, the value of $P(T_i)$ diminishes much faster owing to its gaussian nature. Therefore, the product of P(T_i) and NBTI(T_i) vanishes at both very low and very high temperatures. Considering this, Eqn 3.10 can be simplified by approximating the *erfc* functions to obtain:

$$Degr(i) = \left(a\mu_i^2 + a\sigma_i^2 + b\mu_i + c\right) \tag{3.11}$$

The effective temperature of cell i, T_i^{eff}, can now be defined as the temperature at which if the cell was operating throughout, produces NBTI degradation equal to $Degr(i)$. Symbolically, $T_{eff}^{i} = \text{NBTI}^{-1}\left(Degr\left(i\right)\right)$ where $\text{NBTI}^{-1}(\cdot)$ is the inverse function of NBTI$(\cdot)$ of Eqn 3.9. Solving this, the value of T_i^{eff} is:

$$T_i^{eff} = \frac{-b + \sqrt{\left(b + 2a\mu_i\right)^2 + 4a^2\sigma_i^2}}{2a} \tag{3.12}$$

Plugging the values of a, b, and c from the fit obtained in Section 3.3.2, we obtain:

$$T_i^{eff} = 248.23 + \sqrt{\mu_i^2 + \sigma_i^2 - 496.46 \times \mu + 61619} \tag{3.13}$$

Eqn 3.13 is the closed form expression for computing the effective temperature which should be considered for NBTI degradation calculation for a block whose temporal temperature variation can be approximated by a gaussian with mean temperature of μ_i and the standard deviation σ_i.

Example 3. Consider a block with gaussian temperature variation about a mean value of 340K and standard deviation of 40K. For the purpose of NBTI calculation, the T^{eff} of this block is given by Eqn 3.13 as 348K. Simply considering the mean value instead of T^{eff} would cause 14% error in degradation computation.

To ascertain the accuracy of Eqn 3.13, we computed the NBTI degradation of a cell using Eqn 3.13 as well as through sequential calculation using first principles for gaussian random temperature values. A difference of only

0.2% was observed thus proving that for the purpose of NBTI calculation, a single value of T^{eff} can be used with insignificant loss of accuracy.

3.3.4 Skew Computation Methodology

The skew computation methodology proposed by us is as follows. From RTL simulation of the design, the fraction of time each clock buffer remain gated is extracted. Next, the T^{eff} of each clock buffer is computed (see Section 3.3.3) based on placement of the cells in layout. The next step is to generate the spice netlist of the clock tree of the design. The T^{eff} and gating fraction (GF) of each clock buffer are now used to compute its ΔV_{th} due to NBTI. We used the cycle-by-cycle method for ΔV_{th} calculation detailed in[93]. Each clock buffer in the spice netlist of clock tree is then annotated with its T^{eff} and ΔV_{th} value and a spice simulation is run to compute the clock skew. Our proposed methodology is graphically depicted in Figure 3.14.

Annotating individual T^{eff} and ΔV_{th} for each clock buffer which are part of a single spice netlist is non-trivial. A spice netlist can only take one global operating temperature (using *.option temp*) for a single simulation. One option is to generate a look-up table for each device in isolation with its correct temperature and use these numbers to find clock skew. However, this can be erroneous due to mismatch in driver/loading strengths that may exist w.r.t. the actual clock tree elements. To overcome this issue, we used the following technique: A unique copy of PMOS model was created for each degraded PMOS device and the parameter *.tnom* (which is the device parameter

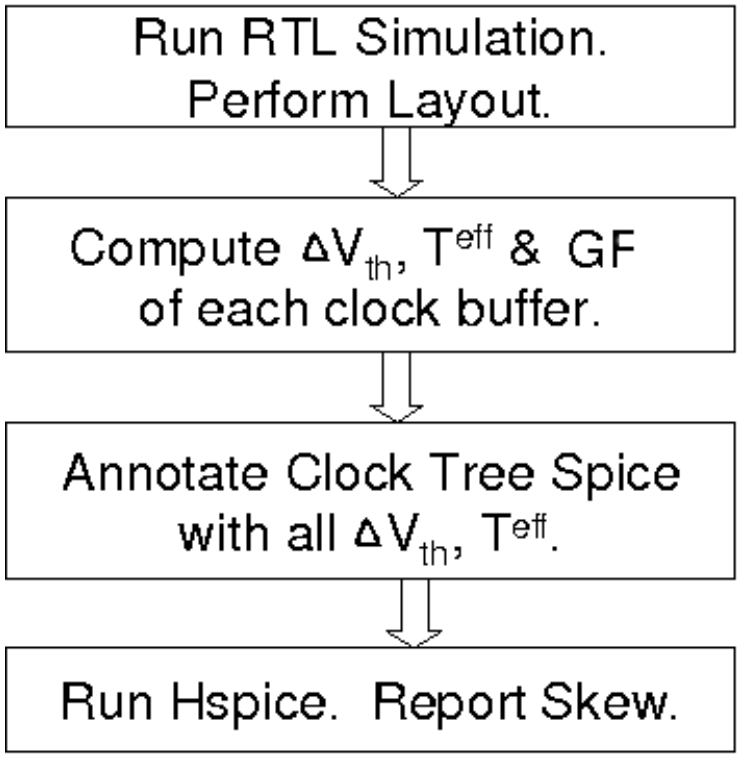

Figure 3.14: Our proposed Temperature Aware NBTI induced skew calculation flow.

extraction temperature) was modified for them such that for a single value of chip operating temperature (using *.option temp*), the interpolation done by spice from the *modified* extraction temperatures would match with the simulation setting we want to enforce. For the sake of simplicity, we used a linear extrapolation for *.tnom* modification however complex dependencies can also be handled easily this way.

Example 4. Consider two clock elements A and B with T^{eff} value of 343K (70^{o}C) and 358K (85^{o}C) respectively. Let the extraction temperature of the P/NMOS

devices be 300K (27^oC). If the operating temperature of the spice deck is set at 343K, HSpice program will interpolate the device parameter from 300K to this temperature. This will be inaccurate for device B because it is operating at a higher temperature than 343K. Using our technique, we make a copy of the device model file and change its extraction temperature to 285K and bind cell B to this model. HSpice program will now interpolate (to 343K) the model parameters of device A from 300K and for device B from 285K thus considering a 15K extra interpolation for device B.

3.3.5 Clock Skew Results

We created the layout of several benchmark circuits from *opencores.org*[12] including clock tree generation with SOC Encounter using $45nm$ open-source *nangate* technology library[11] and passed it through the flow in Figure 3.14. The details of the benchmark circuits are in Table 3.4.

The above benchmarks vary in size from 3K to nearly half a million gates. Except for the couple of largest benchmark, rest are much smaller than typical industrial design, therefore the temperature increase we obtained using thermal simulation was very small. To capture the effect of our technique, we linearly scaled the temperature values obtained through T^{eff} computation to a spectrum of temperature between 50^oC to 110^0C to mimic a large industrial design[60] with significant thermal gradient.

Table 3.5 shows the temperature aware NBTI induced clock skew degradation for our benchmark circuits. For each benchmark, four skew values are

reported. Colm A presents the skew reported by the clock tree generator which is unaware of the temperature gradient. Colm B shows the skew value after considering impact of temperature on delay of the buffers. Colm C and D show the clock skew after including NBTI as well as temperature considerations after 5 and 10 years of circuit aging respectively. Comparison of Colm A and Colm B shows that due to *just* temperature gradients, the skew of clock tree can be very different from what is reported by the clock tree synthesis tool. However, much more troublesome problem is evident by comparison of Colm A and Colm C or Colm D. The effect of NBTI drastically increases the clock skew due to clock gating and temperature dependent NBTI effect. In particular, the clock skew can become as much as 7X compared to the skew reported by clock tree synthesis tool (Colm A vs Colm D, DES^* benchmark after 10 year aging) and even after considering the temperature profile (i.e. Colm B), the actual skew value can be up-to 3X larger. This brings home the point that previously proposed technique in [38] of over-constraining the clock tree synthesis tool by artificially tightening the clock skew constraint cannot be relied on since there might not be enough margin available in original clock skew budget itself.

With the above results in mind, in the next section we will propose our solution to efficiently deal with the NBTI menace to manage the skew of a clock tree over the lifetime of the design.

3.3.6 Proposed Scheme: GBLV

Though NBTI degrades each clock buffer, the clock *skew* is primarily impacted due to *difference* in degradation of individual clock buffers. There are two sources for this difference: a) non-uniform temperatures, and b) non-uniform gating ratio (GR) among different clock buffers. Since NBTI impact is dependent on both these factors, significant differences among different clock buffers can accumulate leading to increased clock skew. In the post-placement stage, little can be done to eliminate thermal gradients, however equalizing GRs of different clock buffers is still possible. To exploit this optimization knob, we propose our scheme called **G**ating with **B**oth **L**ogic **V**alue (GBLV) scheme.

> GBLV Scheme: When gating a clock sub-tree, choose between logic 0 and logic 1 to apply to the frozen clock tree with the aim of equalizing the times these two choices are made.

Each clock buffer is internally made up of two inverters in series. Freezing the clock tree with logic 1 provides recovery (stress) time to the first (second) inverter in each of the gated clock buffer[6]. On the other hand, freezing the clock tree with logic 0 has the opposite effect. We will refer to these two scenarios as 1-Gating and 0-Gating respectively. The ability to choose between

[6]The reasoning remains the same if the clock tree is implemented with inverters instead of buffers. In that case the inverters at even(odd) distance from the clock gating element experiences recovery(stress) time

1-Gating and 0-Gating allows balancing of NBTI degradation thus reducing clock skew.

3.3.7 Hardware Implementation

GBLV technique requires that each clock gating cell should have the ability to freeze its output at logic value of 1 or 0 instead of traditional method of one of them. Figure 3.15 shows a simple optimized design of such a clock gating cell.

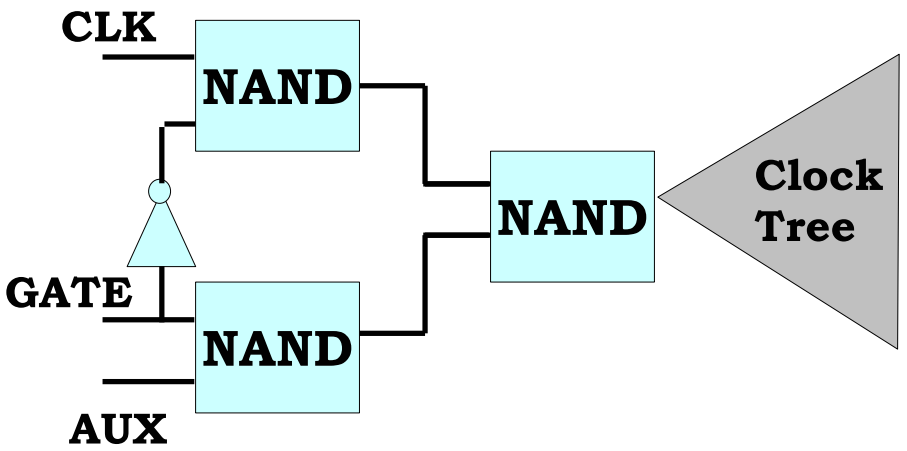

Figure 3.15: Gate-level implementation of optimized 0/1 choosing clock gating element.

When the GATE signal is low, the CLK signal passes through and the clock tree functions as usual. When GATE signal is high (which signals a clock gating opportunity), the clock tree is frozen with the logic value corresponding to signal AUX. Thus, signal AUX provides a handle to implement GBLV scheme.

Clock Gating Cell Overhead: We compared the overhead of our gating cell w.r.t. traditional AND gate based implementation [21] using our $45nm$ library [11]. For a clock frequency of 1GHz, 10% slew, and fanout load of 10 flops, the penalty in leakage power, dynamic power and area is 29.4% (2.2μW vs 1.7μW), 10.5% (25.2μW vs 22.8μW), and 33% (2.12μm^2 vs 1.60μm^2) respectively. However, one must realize that even with fine-grain clock gating, only a fraction of clock buffers are converted as clock gating elements. A typical 1 million cell high performance design with logic depth of 6 may have 160K flops requiring 10K clock buffers. If 10% of these are clock gating elements, the overhead in

- Leakage Power : 0.03%
- Dynamic Power : 0.01%
- Area : 0.01%

3.3.8 Generating AUX Signal

Generating the *perfect* AUX signal that equalizes duration of 1-Gating and 0-Gating in hardware is impractical because it would require complete knowledge of scheduling and duration of all clock gating activities. Except for a few applications (such as GPU, DSP) this information is seldom possible to obtain. It is possible to easily implement the balancing scheme in software with some performance overhead, however, we will focus our attention to hardware

schemes to minimize performance overhead. The AUX signal can be generated using the following two schemes:

1. **Alternate 0 and 1**: If we choose to use 1(0)-Gating when a module is being clock gated, the next time this module has to be gated, we should use 0(1)-Gating. This scheme can be realized using a 1-bit memory near the gating element or using a Toggle flop. There is some area penalty in this method, however there is no routing overhead.

2. **AUX as slow clock**: The AUX signal can itself be a very slow clock signal (frequency = 0.01Hz) thus flipping its value periodically. The low frequency is sufficient to control long-term NBTI degradation and desirable for near-zero switching power penalty. There is routing overhead using this scheme, but with no power/cell area overhead.

The above two schemes have mutually exclusive strengths. In case the duration for which clock gating is enabled each time is similar but the schedule of clock gating is not, the first scheme works perfectly. This is because alternating stress recovery cycles of equal duration will be applied to each clock buffer. On the other hand, if clock gating is enabled regularly but for unequal duration each time, the second scheme works perfectly. This is because the regular clock gating will align with the very slow changing high/low value of AUX providing the necessary balance of stress and recovery phases.

Example 5. A property like "Whenever clock gating is used, its for 1500 cycles" is perfect for the first scheme.

Example 6. A property like "In every 10 minutes of operation, the clock is gated for a total of 90 seconds" is perfect for second scheme.

A designer can choose between the above two schemes depending upon the type of clock gating expected in the design.

3.3.9 Results

By completely eliminating the imbalance between 0-gating and 1-gating, reduced clock skew can be achieved. However, due to the statistical nature of our technique, there might be cases (such as when none of the statistical properties in above examples can be asserted) when the 0-gating and 1-gating do not properly balance. Let the imbalance between 0-gating and 1-gating be described by *mismatch factor* β, which is computed as:

$$\beta = \frac{(T_0 - T_1)}{(T_0 + T_1)} \tag{3.14}$$

where $T_0(T_1)$ is the duration of time when 0(1)-Gating is used. β=0 implies perfect balance of 0 and 1-Gating and higher magnitude of β implies more imbalance between these two quantities. Table 3.6 shows the reduction in clock skew obtained by our method. Colm (orig) and (degr) contain the temperature and NBTI aware clock skew value at the beginning and after 10 year lifetime of the circuit respectively. The next three columns denote the clock skew values for three cases: β=0, β=0.1 and β=0.2 for the benchmarks. Because of space constraints, we are showing the numbers for 10 years aging but the trend for skew reduction is similar 5 year aging of devices.

From Table 3.6, we note that our method of balancing 0 and 1-Gating achieves very good results. The improvements achieved in clock skew degradation (%) when using *GBLV* is graphically represented in Figure 3.16. For the case of perfect balancing, the skew degradation has been reduced by up-to 70% (47% average) after 10 years of aging.

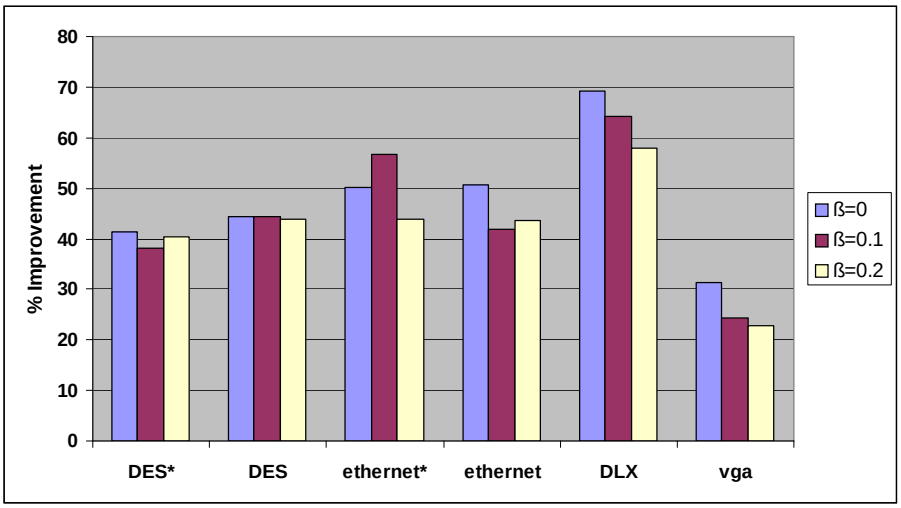

Figure 3.16: Reduction in skew degradation achieved by *GBLV* on 10 year aged benchmarks for different β

For the case of δ=0.1 and δ=0.2 which signify that due to statistical nature of our technique some (10% and 20%) amount of imbalance of 0-Gating and 1-Gating remains, the average reduction in skew degradation is 45% and 42%. This significant reduction in the clock skew can be harnessed to design high performance systems (by shrinking the clock period) or to increase the device lifetime if it is constrained by NBTI affected clock skew violation. As shown earlier, the power and area overhead of using clock gating cell which

can freeze clock trees to a choice of logic 0/1 levels introduce miniscule power and area penalty.

3.3.10 Discussions

In this section, we introduced the concept of effective temperature which significantly simplifies NBTI degradation computation with minute error ($< 0.2\%$). Using this, we presented the first systematic methodology to compute coupled NBTI and temperature induced clock skew of an aging design. The results have highlighted the previously discounted menace of NBTI effects in gating enabled clock trees - the clock skew can increase by up-to 7X if not corrected explicitly. To ameliorate this problem, we proposed a solution in the form of **G**ating with **B**oth **L**ogic **V**alue (GBLV) scheme which can freeze a clock sub-tree at high or low logic value thus ensuring a balance between stress/recovery cycles of all clock buffers. Use of GBLV can reduce the NBTI induced clock skew degradation by up-to 70% (47% average) with very insignificant ($<0.03\%$) increase in leakage power of the design, dynamic power and area. Two different implementations of GBLV schemes were discussed, each targeting a different statistical property of clock gating. Our technique also showed strong results (44% reduction in clock skew degradation) when the clock gating behavior deviated from assumed statistic properties.

3.4 Reducing NBTI Aging during Burn-in Testing

In this section we will outline a technique to reduce the impact of NBTI aging during the burn-in testing of a product. Due to increasing defect density and variation during manufacturing process, testing of the fabricated products has becomes very difficult for sub-65nm VLSI technology node. The failure rate of a typical product is famously recognized as bath-tub shaped [62] and is characterized by three phases. The first phase is the infant mortality period where the products with inherent defects or manufacturing aberration quickly fail. This is followed by relatively flat second phase with low failure rate which depicts the normal operation of the product. The third phase occurs near the end of expected lifetime of a product where the wear-out mechanisms cause the product to malfunction or fail altogether. Of particular concerns are the products that belong to the first type where devices appear to operate correctly after fabrication but will fail shortly after being deployed in the field. VLSI testing deals with this problem by performing (aging) accelerated tests. By operating the product under voltage, thermal, or mechanical stress, the devices in it can be artificially aged much quicker than the real duration of operation of testing. Burn-in testing is one such test to screen and eliminate marginal devices and is performed at significantly increased supply voltage and temperature. The aim of burn-in test is to expedite the onset of infant mortality by accelerating the aging of product. The acceleration in aging during burn-in testing is the product of acceleration due to higher supply voltage and the acceleration factor due to higher operating temperature [85].

A functional test done after burn-in testing can catch most of the failures due to ion contamination and resistive interconnects. Recent work in the area of burn-in testing has focused on reduction of burn-in time, and burn-in recipe modification [85] by choosing appropriate values of elevated supply voltage or temperature.

There are two types of burn-in tests: a) static burn-in and b) dynamic burn-in. Static burn-in is performed by operating the product to be tested at elevated voltage and temperature but with a fixed vector applied to its input thus no switching occurs in the circuit. On the other hand, during dynamic burn-in, a series of vectors are applied to the circuit causing switching of various circuit nodes. In a typical burn-in recipe, both static and dynamic burn-in testing is performed. Due to its constant bias nature, static burn-in causes more degradation than dynamic burn-in testing. Through HSpice simulations, we observed 4X larger V_{TH} degradation during static burn-in compared to dynamic burn-in, in line with trend in [17].

One may wrongly assume that since static burn-in testing is performed for a short duration, its impact on V_{TH} increase may be insignificant. Due to the exponential (polynomial) dependence of NBTI on temperature (voltage), we expect significant V_{TH} increase during burn-in test. To quantify, we simulated the V_{TH} increase of two 45nm PMOS devices under regular vs burn-in testing conditions: the first device was simulated at supply voltage of 0.9V, temperature of 65^oC operating for 10 years. The second device was simulated at a supply voltage of 1.25V, temperature of 125^oC operating for

only 10 hours. These set of conditions correspond to aging acceleration factor of approximately 1000. The resultant increase in V_{TH} of the second device was as much as 61% of that of first device. This experiment shows that in just 10 hours of static burn-in, the device accumulates more than 60% of the degradation it would accumulate in 10 years operating in field. Therefore, the NBTI degradation of a chip due to burn-in testing cannot be neglected even if the burn-in duration is short. To the best of our knowledge, there has been no prior work to estimate and reduce the NBTI degradation caused by static or dynamic burn-in testing. Our primary contributions in this work are:

1. We quantify the threshold voltage (V_{TH}) degradation due to NBTI during static burn-in testing and show that severe degradation can result.

2. To reduce the NBTI impact of static burn-in we propose use of minimum NBTI induced delay degrading vector (MDDV) during burn-in.

3. We formulate the problem of finding `MDDV` as a zero-one linear program (ZOLP) that automatically considers timing criticality and NBTI sensitivity of cells in the circuit.

3.4.1 Reducing NBTI in Static Burn-in

During static burn-in testing, a static (i.e. non changing) vector is applied at circuit input which biases the circuit in a given fixed configuration under high operating temperature and voltage. Due to the presence of sequential elements (flops or latches), it is not enough to only determine the logic

values at each primary input of the circuit since this will bias only those gates which lie on paths from primary inputs to first set of sequential gates. Therefore, the input vector assignment should assign logic values to the output of the sequential elements apart from the circuit primary inputs. For testability reasons, all the sequential elements in the circuit are chained together using scan chain methodology through which the required input vector assignment even deep inside the circuit can be performed. We assume that the design independent environmental parameters of burn-in testing recipe are at their optimal value depending on analysis of failure rate decrease and thus cannot be modified. These parameters include the burn-in duration, voltage and temperature whose values are determined after failure rate analysis as well as aging acceleration requirements.

3.4.2 Minimum Delay Degrading Vector (MDDV)

The choice of the input vector applied during the static burn-in test presents itself as a strong design dependent optimization knob. This is due to different delay sensitivity of various gates to V_{TH} degradation and logic value propagation due to functionality implemented by the gates. A good input vector during burn-in testing can distribute the V_{TH} degradation among the gates such that the degradation of some figure of merit of the design is minimized. Since NBTI mainly impacts performance (and signal integrity indirectly), we define the critical path delay of the circuit as the metric to optimize. We call this vector as Minimum Delay Degrading Vector (MDDV) and

formally define the problem it as:

Definition 3.4.1. For a given burn-in recipe (duration, voltage, and temperature), gate level netlist of the product, and timing characterization of standard cells, find the (possibly partial) input vector which when applied during static burn-in, minimizes the increase in critical path delay of the product.

The idea of using `MDDV` is similar to using minimum leakage power vector (MLV) which is defined as the input vector which causes the least leakage power in the circuit during stand-by stage of the circuit. Some of the representative work discussing exact and heuristic solutions for MLV are [50][47]. Finding MLV is an NP complete problem: exact solutions are typically SAT based and the heuristics proposed are mainly randomized search based. However, the problem of determining `MDDV` is very different from determining minimum leakage vector in many aspects:

- *Cost Function*: The optimization objective of MLV problem is simply the algebraic sum of all leakage power values of individual gates. However, the optimization objective of `MDDV` is to reduce delay through the circuit which depends on relative delay among competing paths thus more difficult to optimize.

- *Solution Type*: In MLV, the complete (i.e. all bits of) the input vector must be computed for obtaining the least leakage. However, due to the presence of non critical cells, it is possible to partially determine some

bits of MDDV vector which feed timing critical cells. This allows surgical analysis of the circuit using high complexity formulations such as zero-one linear program (ZOLP).

- *Scope*: Since the input vector is only partially determined in MDDV, other optimization objectives can be co-optimized to assign logic values of rest of the input pins.

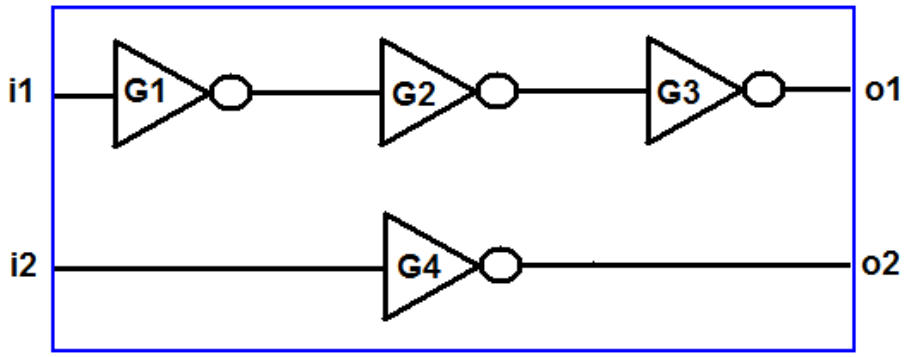

Figure 3.17: Illustration demonstrating partial determination of input vector for the purpose of NBTI induced delay degradation reduction. Non timing critical inputs are ignored if their delay can never increase enough to become critical path.

Example 7. Figure 3.17 shows a toy example circuit depicting the concept of partial vector determination. The circuit has two primary inputs (i1 and i2) and two primary outputs (o1 and o2). Assume that each inverter gate has a nominal delay of 1ps. The critical delay of the circuit is 3ps due to the top path. Now, consider that due to NBTI degradation, delay of each inverter increases by 10%[7]. Thus, the degraded delay of the top path is 3.3ps whereas

[7]In reality delay increase is not a fixed number for different gates, the assumption of

that of the bottom path is 1.1ps. Since the degraded delay of the bottom path is lesser than the original critical path delay of the circuit, we infer that NBTI degradation can never make that path critical, hence we ignore this path and focus on only the top path. For minimum increase in the delay of the top path, the input i1 must be set to logic 1. This finishes the determination of `MDDV` by forcing only partial bits of input vector to desired value (i.e. i1 = 1). The other input (i.e. i2) are don't cares for NBTI degradation analysis and can be set to any value depending on any simultaneous optimization objective such as leakage power.

3.4.3 `MDDV` Determination Algorithm

Algorithm 3 shows the overall flow of our algorithm for finding `MDDV`. Each step of the flow is described later in this section. The first step in our flow is to determine all probable cells, the logic value of whose pins need to be controlled for reducing NBTI induced delay degradation. These cells comprises of the fanin cone of any timing endpoint (i.e. primary outputs or inputs of sequential cells) that do not meet the timing constraint after NBTI degradation. For each net driven by the output pins of such cells, we assign a binary variable which represents its logic value during static burn-in testing. Depending on the logic implemented by each cell, logic equations among the variables associated with the input and output nets of a cell can

10% is only for illustration. For the sake of simplicity, in this example the difference in the dependence of rise and fall delay is ignored.

be determined. Similarly, timing constraints among the input and output nets can be obtained by assigning arrival time variables which allow critical path optimization with linear number of constraints [56]. With these relations, the problem of determining the minimum NBTI degradation vector `MDDV` can be cast as a zero-one linear program (ZOLP).

Algorithm 3 Compute `MDDV` vector for static burn-in

1: Determine post static burn-in critical cells
2: Assign binary variable to net connecting critical cells
3: Determine logic implication and timing constraints
4: Cast `MDDV` finding as ZOLP problem

Determining post burn-in critical cells: Algorithm 4 describes method we used to determine post burn-in critical cells. With the understanding that burn-in will degrade the circuit timing to some extent, the algorithm takes an *optional* parameter (default value = 0) called acceptable degradation (AD) which denotes the degradation that is acceptable during burn-in testing (Line 2). The required time of the circuit that is used during timing analysis is incremented by the acceptable delay degradation (Line 3). The goal of providing a realistic number for acceptable degradation is to reduce the computation efforts by removing the paths whose increased delay will be less than AD even if the PMOS devices in it undergoes NBTI degradation. With the updated value of required time, static timing analysis (STA) is performed to identify timing failing paths (Line 5). During this STA run, we load the degraded timing library which has delay numbers corresponding to behavior of a cell after a particular threshold voltage increase has occurred. From the

results of STA, all the endpoints which do not meet the timing are identified. For each of these endpoints, we find the cells lying in their fanin cone. All such cells are marked as critical cells.

Algorithm 4 Find post-burn critical cells

1: {// Modify required timing}
2: Input: Acceptable Degradation AD%
3: Increase Required Arrival time by AD%
4: {// Run STA with new library}
5: Load timing information from V_{TH} degraded library
6: Perform STA and find timing violating endpoints (EP)
7: SEL = Φ
8: **for all** ep in EP **do**
9: SEL = SEL $\cup$ cells in fanin cone of ep
10: Return SEL

Input-Output logic relation for each cell type: The output net of each critical cell identified in Algorithm 4 is associated with a binary variable which represents its logic value during static burn-in testing. We refer to these nets as critical nets. Depending on functionality of the cell, there exists a logic relation between these variables. Let the input pins of cell x be denoted by IN(x) and the output pin by OU(x). Further, for such a cell, let the relationship between the logic values of these pins be given by

$$OU(x) = F(IN(x)) \tag{3.15}$$

For example if the binary variable associated with the input and output of an inverter gate is x_a and x_z, then they are related by relation $x_z = 1 - x_a$. Consider an OR gate whose two input pins are associated with binary

variables x_a and x_b and the output pin is associated with binary variable x_z. These variables are connected through the relationship in binary form as $x_z = x_a + x_b - x_a x_b$. Similarly, for other cells, the relation binding the input and output binary variables are established.

Generating delay constraints: We associate a real valued variable for each of the critical nets. The value of this variable represents the arrival time of the signal *after NBTI degradation of the circuit*. Similar to method shown in [56][18], we generate linear constraints relating the signal arrival time at input and output of the cells to minimize the critical path delay of the circuit. Basically each of these constraints ties the signal arrival time of the output of any cell to be larger than or equal to the sum of signal arrival time at any of its input and delay of the timing arc from the corresponding input to the output. Let the delay of the timing arc from input pin i to the output pin o be D_{ij}^O originally (i.e. without NBTI degradation of the devices in the cell) and D_{ij}^N after NBTI degradation. Further, let the logic value of net driving pin i be x_i, then the timing arc delay can be written as

$$D_{ij} = x_i \times D_{ij}^O + (1 - x_i) \times D_{ij}^N \tag{3.16}$$

that is, when the cell is driven by logic high (i.e. $x_i = 1$) during static burn-in testing and there is no NBTI degradation in its PMOS devices, D_{ij} equals D_{ij}^O. In case of a gate driven by logic low (thus undergoing NBTI degradation), D_{ij} equals D_{ij}^N. Though NBTI impacts only the PMOS devices, it may seem enough to consider only the rise time of a cell. However, the timing arcs

from an input pin to output pin can be either negative unate (e.g. NAND gate), positive unate (e.g. AND gate), or even non-unate (e.g. XOR gate). Therefore, both the rise and fall delay of every timing arc needs to be written in the form of Eqn 3.16.

ZOLP Formulation to determine MDDV: Once the boolean logic relation among the binary variables associated with the input and output pins of each cell is ascertained, we can formulate the problem of finding `MDDV` as a zero-one linear program (ZOLP). Let *EP* be the set of timing endpoints (i.e. D pins of flops or primary outputs) that are driven by the critical cells identified. These are the endpoints whose timing slack is negative and therefore their arrival time needs to be optimized simultaneously. The arrival time of the signal at a node x is denoted as AT(x). Further, let all the cells selected to be critical be represented by *SEL*. The input pins of cell x will be denoted by IN(x) and without loss of generality, we assume that all cells have single output pin denoted by OU(x). The formulation for finding `MDDV` is:

$$
\begin{aligned}
Minimize : MAX & \\
Subject\ To : & \\
AT(i) \leq MAX & \qquad \forall i \in \mathbf{EP} \\
AT(i) + D_{ij} \leq AT(j) & \quad \forall i \in IN(x) \quad \forall j \in OU(x) \quad \forall x \in \mathbf{SEL} \\
F(IN(i), OU(i)) = 1 & \qquad \forall i \in \mathbf{SEL} \\
x_i \in \{0, 1\} & \qquad \forall i \in \mathbf{SEL}
\end{aligned}
$$

By requiring all endpoint delays to be less than the dummy variable

MAX in the first set of constraints, and minimizing MAX, we can effectively obtain the logic value assignment for nets connected to the critical cells so as to obtain the fastest possible circuit. The second set of constraints establishes the actual delay relationship between input and outputs of a cell where the timing arc delay from input pin i to output pin j is given by Equation 3.16. The third set of constraints are for relationship between logic value for input and output pins for each critical cell based on Eqn 3.15. Note that depending on these equation, there will be multiplication of variables in the above formulation. Thus it may seem that the constraints in the above formulation is not linear but quadratic in nature. However, by virtue of the variables being binary, a multiplication of any number of variables can be decomposed in terms of linear equations [31]. For example, consider the term $\mathrm{x}_A\mathrm{x}_B$ which is multiplication of the binary values x_A and x_B. We can replace the multiplication the variable $\mathrm{x}_A\mathrm{x}_B$ by a new binary variable M_{AB} and add the following constraints to the above formulation:

$$\begin{array}{lcl} x_A + x_B & \leq & 1 + M_{AB} \\ (1 - x_A) + (1 - x_B) & \leq & 2 - 2 \times M_{AB} \end{array}$$

It can be verified that the only condition under which the value of binary variable M_{AB} is 1, is if both x_A and x_B are 1. The above transformation can be repeatedly applied thus allowing conversion of arbitrary multiplication of binary variables to linear equations. Once the above ZOLP is solved, we can obtain self consistent logic values on each pin of the critical cells identified. The solution to the above ZOLP also determines the logic values at the inputs that drive the timing failing paths. In case of sequential circuits, the inputs of

the circuit comprises of primary inputs as well as the output pin of sequential elements.

ZOLP Speedup: Depending on the histogram of slack at the timing endpoints of a circuit, there could be a large number of timing violating endpoints for some circuits. Solving one large ZOLP for such circuits can be infeasible due to runtime issues. For such cases, the critical cells can be naturally partitioned into independent components and each such component can be solved independently. The failing cells can be partitioned into independent sets by traversing the fanin-cone of each timing failing endpoint going backwards towards inputs. If during the traversal from an endpoint ep_a, we visit a cell already visited by traversal of endpoint ep_b, the search proceeds after merging the two sets of cells from the traversal of the two endpoints. In our experiments, we *always* identify independent components and solve them individually.

3.4.4 Experimental Setup

Our complete flow was implemented in C++ language with the gate level simulations being done using HSpice [89] tool. All experiments were run on an Intel dual core 2.66 GHz workstation with 4GB main memory running Debian OS. To enable post burn-in static timing analysis (STA), we created a new liberty (.lib) file containing the timing information for the standard cells in degraded state. Static timing analysis to identify timing failing cells was done by Primetime [89] tool. We used open-source Nangate [11] 45nm

library for all experiments with nominal PMOS V_{TH} of 350mV. The number of standard cells in this library is 134. The magnitude of V_{TH} degradation in case a PMOS gate undergoes NBTI degradation during the entire duration of static burn-in (i.e. is driven by logic low) was found to be 42mV based on our NBTI simulation. For our implementation we characterized the degraded delay of only those cells which actually appear in the circuit netlist. To solve the ZOLP formulation for finding `MDDV`, we used the MILP solver Gurobi [51].

Table 3.7 shows the benchmarks used in our work from the open-source repository [12]. *LDPC* is the decoder unit of Low-Density Parity Checker. *wbConmax* is a wishbone interconnect matrix IP. Benchmark *ethernet* is an ethernet wishbone interface. *DES* is a data encryption standard core. The second column onwards contains the number of cells, nets, number of IO pins, flops, number of primary inputs, and number of primary outputs in the circuit. For these benchmarks, we solved the `MDDV` problem to obtain the best partial input vector. The delay degradation when `MDDV` is applied during static burn-in testing will be compared to that when applying either all 0s or all 1s as the input vector. To substantiate the claim that only some of the bits of the input vector needs to be specified, we also compute the fraction of the input pins whose value is determined by `MDDV`. Note that the logic value of both circuit *primary inputs* as well as output pins of the flip flop output are under purview of input vector determination. In our experiments, the acceptable delay degradation due to NBTI is set to 2% of the critical delay, thus during STA using degraded timing numbers, any path whose delay is more than 102% of critical

path delay before burn-in testing was marked as failing timing requirement.

3.4.5 Results

The results of computation of MDDV is shown in Table 3.8. For each benchmark, column *OrigDelay* shows the critical path delay (critical delay here onwards) in picoseconds (ps) before starting the burn-in testing as reported by STA using the original (i.e. non degraded) library. The degradation due to three input vectors are shown next: a) input vector comprising of all bits at logic low (*All 0* in the Table), b) input vector comprising of all bits at logic high (*All 1* in the Table), and c) the MDDV vector. For each of these three settings of input vectors, the delay achieved after static burn-in testing appears in column *AchvDel* (in picoseconds) and its degradation w.r.t. the original critical delay is shown in column $\Delta Delay$. The column *CPU Time* in the table shows the CPU runtime (in seconds) for preparing and solving the ZOLP formulation for a given benchmark. This runtime includes the time for data reading, STA, critical cell finding, and ZOLP generation and solving. The last column *#BinVar* shows the number of binary variables in the generated ZOLP. The time required to characterize the library for a given V_{TH} is not reported since it is a one time effort and the resultant library can be used for many designs.

From Table 3.8, we observe that the MDDV vector can be computed in reasonable time for all the above circuits. The delay degradation of the circuit after burn-in testing when using the input vector comprising of all bits at

logic low (high) is observed to be 7% (7.2%). This corroborates our thesis that burn-in can significantly degrade a circuit due to the NBTI effect. For example, if speed binning is performed on a 2.66 GHz processor after the burn-in conditions simulated in our work, it will run at 2.33 GHz potentially ending up performing at speeds lower than its capability. Intuitively, since NBTI occurs in PMOS devices at negative bias, we expected an all 0 input vector to give much higher degradation than all 1 input vector. However, the degradation due to these two vectors is nearly the same. On further analysis we found that the reason for this observation is that because of logic functions implemented by different gates, an all 0 input vector though stresses the gates close to the launching flops/primary inputs, may induce a logic high value at some other internal node thus reducing its degradation. On application of the NBTI induced minimum delay degradation vector (`MDDV`), the circuit delay degradation is reduced to only 3.3%. Therefore, using the `MDDV` vector during static burn-in testing can reduce the delay degradation by more than 2X.

Figure 3.18 shows the fraction of pins whose value is forced by the solution of `MDDV` determination compared to the total number of assignable pins in the circuit. Note that each primary input as well as flop output pin are assignable. From the numbers in this figure, the first observation we make is that the fraction of pins assigned can vary widely among different circuits. Due to the highly convoluted interconnect structure of `LDPC` decoder, more than 50% input pins get assigned as part of solving `MDDV`. On the other hand, due to the unique structure of *ethernet* (notice that 25% of cells are sequential

elements from Table 3.8), different cells are not highly connected. The number of input pins that MDDV enforces was found to be just 1.1% for this benchmark. Overall, on an average only 25% of the input pins are forced by MDDV and the logic value for the rest 75% of the pins can be chosen to optimize a secondary objective during burn-in testing such as leakage power thus reducing power dissipation as well as heating of the chip due to it. Data for figure:

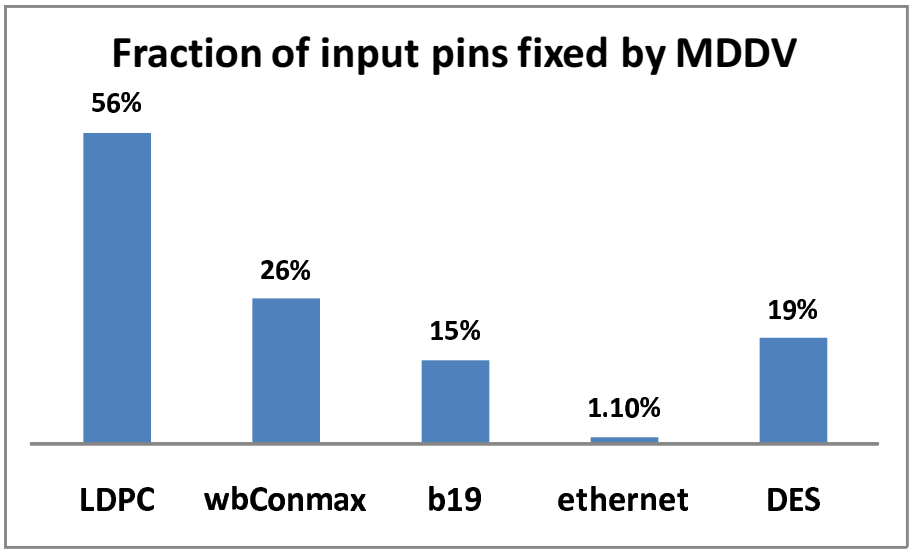

Figure 3.18: Fraction of input pins whose logic value is fixed by MDDV vector. Rest of the pins are available for optimizing secondary objective.

3.4.6 Discussions

In this section, we quantified the negative bias temperature instability (NBTI) induced threshold voltage (V_{TH}) increase during the burn-in accelerated testing process. The problem of reducing the impact on critical path delay increase due to NBTI was targeted by identifying a minimum NBTI

degradation vector (MDDV). A good MDDV distributes the V_{TH} degradation in an intelligent way considering the timing criticality and NBTI induced V_{TH} increase sensitivity of different cells. Computation of MDDV was formulated as a zero-one linear program (ZOLP). Thanks to non-timing critical paths in the circuit, the MDDV vector only requires a subset of input pins to be at a pre-defined logic values which allows optimization of other objective such as leakage power using other input pins. On a set of benchmarks, our technique reduces the critical path delay degradation by more than 2X while requiring only 25% of the input pins to a fixed logic value. Our future work would be in the area of burn-in recipe (duration, voltage, temperature) optimization for NBTI reduction.

Table 3.3: Clock skew achieved by our solution is compared with all-NAND, all-NOR, and random best-among-10-trials strategies. Data preparation time and solver time are reported in seconds. All "Penalty" columns show the extent of penalty (in %) of using the corresponding technique instead of the proposed optimal solution.

Name	Solver CPU(s)	% of NAND	% of NOR	OUR Skew(ps)	∀ NAND Skew(ps)	Penalty %	∀ NOR Skew(ps)	Penalty %	Rand* Skew(ps)	Penalty %
A	0.14	77%	23%	2.80	4.41	57.50%	9.02	299.03%	7.24	158.57%
B	0.06	97%	3%	2.18	3.23	48.26%	5.84	167.28%	4.96	125.68%
C	1.41	71%	29%	4.13	6.64	56.14%	9.28	124.69%	7.05	70.70%
D	0.81	81%	19%	3.03	5.04	37.81%	9.74	221.45%	6.21	104.95%
E	0.12	73%	27%	2.76	5.46	66.33%	10.21	269.92%	7.04	225.92%
F	0.09	60%	40%	3.94	6.21	57.61%	12.23	210.40%	11.82	200.00%
G	0.47	77%	23%	3.88	6.75	73.94%	13.07	237.11%	10.58	172.84%
H	0.09	83%	17%	2.59	3.91	50.95%	8.44	225.86%	5.38	107.72%
Avg.		77%	23%			56.07%		219.45%		133.75%

*Best result chosen among 10 random tries.

Table 3.4: Benchmarks (from [12]) used in this work.

Name	Num Cells	Num Nets	Num Flops	Num Clk Buf	Area μm^2
DES*	441K	453K	53K	5K	1168K
DES	103K	105K	8.8K	1.2K	198K
ethernet*	238K	242K	53K	4.3K	884K
ethernet	48K	49K	11K	998	180K
DLX	14K	15K	1.6K	174	38K
vga	3.2K	4K	702	65	14K

* means several (6) copies were simultaneously placed.

Table 3.5: Clock Skew values (in *ps*) obtained with: A-Temperature unaware; B-Temp aware; C-Temp + NBTI aware (after 5 years); D-Temp + NBTI aware (after 10 years) delay computation.

Name	Skew A	Skew B	Skew C	Skew D
DES*	84.46	246.62	458.55	482.54
DES	117.59	279.23	374.59	392.95
ethernet*	64.86	160.42	231.57	257.48
ethernet	16.83	40.13	86.56	92.19
DLX	29.53	89.39	107.96	118.02
vga	8.05	17.42	58.54	59.21

Table 3.6: Clock Skew (in *ps*) for 10 year aging. (orig) = temperature aware skew; (degr) = temperature+NBTI aware skew without GBLV; last three columns = temperature+NBTI aware skew with GBLV for increasing β.

Name	Skew (orig)	Skew (degr)	Skew (β=0)	Skew (β=0.1)	Skew (β=0.2)
DES*	246.62	482.54	385.21	392.54	387.75
DES	279.23	392.95	342.48	342.43	343.02
ethernet*	160.42	257.48	208.83	202.54	214.99
ethernet	40.13	92.19	65.86	70.33	69.42
DLX	89.39	118.02	98.23	99.64	101.43
vga	17.42	59.21	46.14	49.05	49.65

Table 3.7: Benchmarks used in this work. Number of cells, nets, pins, flops, primary inputs and outputs are shown for each benchmark.

Name	# Cells	# Nets	# Pins	# Flops	# Inp	# Out
LDPC	44K	48K	4100	2048	2051	2049
wbConmax	27K	29K	2546	818	1130	1416
B19	87K	99K	77	6118	47	30
ethernet	42K	43K	210	10544	95	115
DES	56K	59K	298	8808	234	64

Table 3.8: Comparison of delay degradation using `MDDV` compared to all 0 and all 1 vector.

		All 0		All One		MDDV			
Benchmark	OrigDel	AchvDel	ΔDelay	AchvDel	ΔDelay	AchvDel	ΔDelay	CPU Time	# BinVars
LDPC	1.581	1.694	7.14%	1.695	7.26%	1.627	2.90%	635s	4616
wbConmax	1.308	1.411	7.87%	1.398	6.88%	1.347	2.98%	386s	1688
B19	4.646	4.950	6.54%	4.947	7.13%	4.814	3.61%	1024s	6159
ethernet	2.091	2.247	7.46%	2.238	7.04%	2.178	4.16%	158s	380
DES	1.709	1.812	6.02%	1.832	7.25%	1.758	2.86%	327s	1381
Avg:			7.01%		7.10%		3.30%		

Chapter 4

CAD for Regular Fabrics

Skyrocketing (NRE) costs and increasing variability associated with an ASIC design flow and unacceptable power and delay penalty associated with FPGA design flow have forced semiconductor companies to look for alternatives. One viable alternative that has emerged over time is the use of *structured* ASICs. Structured ASICs provide an exciting middle-ground between high performance of ASIC designs and short time-to-market FPGA designs. They exploit the fact that not all mask-layers provide equal value for the customers and these layers can be pre-fabricated amortizing their cost over multiple designs [84]. Structured ASIC flow is much simpler than that for traditional ASICs because majority of deep submicron issues such as signal integrity, power grid optimization, low skew clock tree distribution are already taken care of by the structured ASIC vendors. Structured ASICs can be used to implement designs consisting of millions of gates in contrast to FPGAs which can implement much lesser number of gates. There are a wide variety of structured ASIC architectures, but all of them have a fundamental repeated logic element called *tile* which may contain pre-defined combinational logic, small RAM, and registers [13].

To fully utilize the benefits of structure ASICs, tools for high quality placement and routing need to be developed. Placement for structured ASIC requires cells to be mapped exactly on a compatible site, similar to the case for FPGA. However, since the problem size of structured ASIC can be an order of magnitude bigger than FPGA [84], the existing FPGA tools cannot be used for structured ASICs. Not only does the placer have to handle millions of cells along with the site compatibility requirement, the task is made much more difficult due to the clocking schemes in structured ASICs. The clocks are built with pre-allocated resources to provide low skew clocking and simplified design flow. This restricts the number of clocked elements which can be placed in proximity of each other. In this work we present RegPlace, a high quality open source placement tool for structured ASICs which can deal with the above mentioned legality and clock constraints.

Higher power consumption compared to the ASIC approach can prove to be the Achilles heel of structured ASIC methodology. To reduce the power penalty, there have been many innovations in their architecture including use of via programming (instead of SRAMs which drain leakage power), low leakage triple oxide devices, and power-down of unused components. From the EDA perspective, the power savings can be increased if the design tools can exploit these power optimization methods. One such step is to enhance placement algorithms to maximize the unused components so that they can be powered down. Distribution of cells as decided by placement determines which areas of the structured ASICs must remain powered on. Due to limited granularity of

power down available, many non functioning devices may be forced to remain powered up due to neighboring functioning devices, leading to wasted leakage power. In addition, placement also determines the extent of the clock network that must switch at each clock cycle. Typically, cells belonging to only one clock group can be placed close to each other, therefore wrong distribution of cells can significantly burn much more power by requiring larger clock network to remain active. We tackle power aware placement on structured ASIC platforms for the first time and propose methods to reduce the leakage and clock power.

The rest of this chapter is organized as follows: We describe the intricacies of placement for structured ASIC in Section 4.1. Section 4.2 describes our key innovations and the overall placement flow. We present a new algorithm for reducing leakage and clock power dissipation of structured ASICs in Section 4.3.

4.1 Introduction to Structured ASICs

There are four types of cells in Nextreme line of structured ASICs. They are SRAM programmable Logic cells (LCELLS), flip flops (DFF), registers (REG) and memories (BRAM). In the rest of this chapter, we will refer to these types of cells with their short names and all these types will collectively referred to as *cells*. The placement problem requires that a cell can only be placed on a location which is reserved for that type of cell. In the basic repeating *tile* of Nextreme architecture the space reserved for LCELL, DFF,

REG and BRAM are as shown in Figure 4.1. There are 36 LCELL and 24 DFF columns in each tile. Each of these column accommodate 64 LCELLs and DFFs respectively. A total of 4 REG and 1 BRAM can be accommodated in each tile. This tile repeats all over the chip with some horizontal and vertical inter-tile whitespace between adjacent tiles. Depending on the size of the netlist to be placed, the number of times these tiles need to be repeated can be configured. Table 4.1 shows the details of two such configured platforms along with the maximum cells of each type that can be accommodated in them.

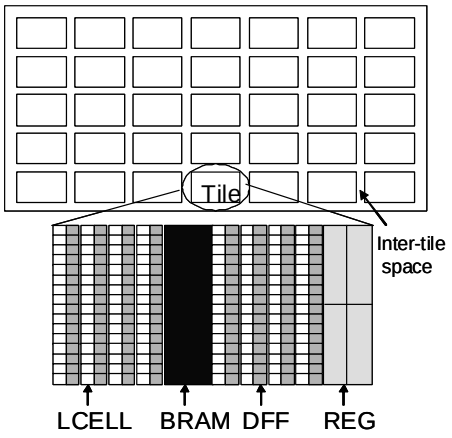

Figure 4.1: Structured ASIC platform.

Table 4.1: Platform Characteristics

Plat.	Max LCELL	Max DFF	Max REG	Max BRAM	Intertile Space	
					Horz	Vert
A	1M	675k	1760	440	8.00	6.45
B	1.2M	811k	2112	528	8.00	8.92

The sizes of each LCELL, DFF, REG and BRAM in the mapped design

netlist are 1x1, 1x1, 8x32 and 16x64. These cells are to be placed according to the reserved column locations described above. The presence of these four different types of cells require that the global placement ensure that the density requirement for each type of cells is met. The presence of whitespace and interleaving of space for each of the cell types makes the physical location available for each type of cell very discrete.

Because of architectural constraints, only a certain number of different clocks can go inside a tile. For example, in Figure 4.1, each tile can have N(=4 in our benchmarks) different clocks. Moreover, all the cells in one column of DFF can have only one clock type. Since REG and BRAM can take more than one clock, their contribution to the maximum number of clocks allowed in tile needs to be taken care of. The clock constraint on placement has big impact on final placement because clock violations may require that the violating cells be reallocated to another region far away from their optimal location.

There are several commercial and academic placers for ASIC designs [82] [34] [94] [70] [26]. These placers implement sophisticated mathematical or min-cut formulations to generate very good quality results for ASIC designs. However, due to site compatibility and hard clock constraints they cannot be directly used for placement of structured ASIC designs. Nevertheless, we note that if the key algorithms in these placers can be somehow used to solve the structured ASIC placement problem, the solutions would be of good quality. Another class of placement tools which are designed for FPGAs can take care of site constraints. However, due to smaller sizes of FPGA based designs,

the existing FPGA placement approaches such as [22][41] rely on slow algorithms like simulated annealing which cannot scale to problem sizes frequently encountered in structured ASIC designs.

There is limited research in the domain of placement for structured ASICs. The work in [69] only addresses incremental placement issues in structured ASICs. Companies are currently using existing row-based placers with product specific legalizers or heuristic [77]. However, there is a dearth of tools which can handle the clock constraints or exploit the modularity of structured ASIC platforms.

Our main technical contributions in this work are

- We propose row-based placer friendly virtual platform generation. This method achieves even density distribution and much faster placement.
- An integer linear program (ILP) formulation is proposed to satisfy clock constraints on the number and type of clocks that can appear in a tile as well as each column of a tile.
- A detail placement flow is proposed specifically for tile based structured ASIC architectures.

4.2 Novel Placement Flow for Structured ASICs

Our complete placement flow is depicted in Figure 4.2. We outline the major steps here while the highlights of each step are discussed in next sections.

In the first phase, given a design netlist and physical platform specification, we transform them into virtual netlist and virtual platform. This key step mitigates the problems arising due to severely discretized space available to the placer. A high quality row-based placer is run to place the virtual netlist on the virtual platform which generates an initial solution for our problem. In the second phase, we transform the placement solution back to the real platform while minimizing the impact on solution quality due to this transformation. The third stage performs the key step of satisfying clock and density requirement at the platform level using efficient mathematical formulation. This is followed by intra-tile clock assignment and perform aggressive wirelength reduction while maintaining the site legality of the solution. The key highlights of the above stages are presented below.

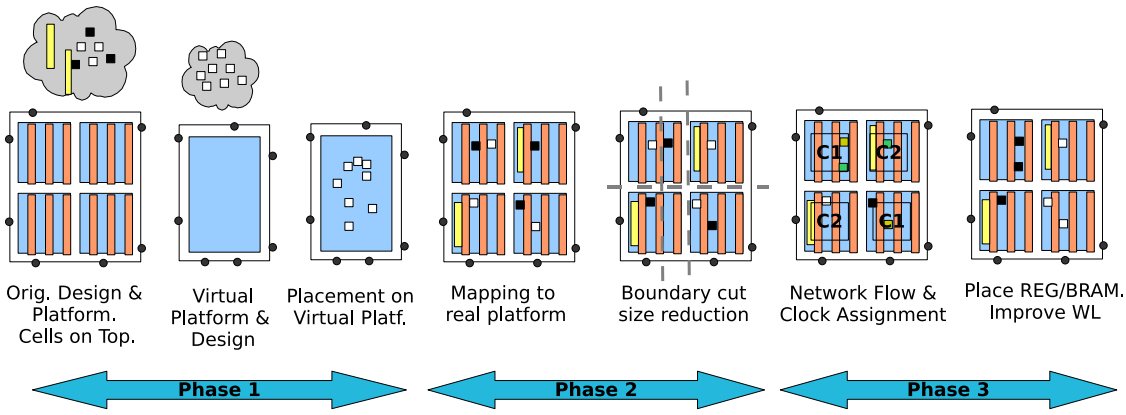

Figure 4.2: Flow of RegPlace. Different colored columns on platform represent different type compatible sites.

4.2.1 Virtual Platform Generation

The physical space available for placement of each type of cell is very discrete in our placement problem. One obvious method to take care of this is through the use of blockages. Though some existing placement tools do have the capability to consider blockages, we cannot specify our placement problem with blockages because blockages for one type of cell can be valid site for another type of cell. Further, the solution quality and the time required by placers degrade significantly when considering several thousands of blockages. To overcome this challenge, we generated a virtual platform and corresponding virtual netlist. The virtual platform is a physically shrunk copy of the real platform with only the reserved space for one type of cell adjacent to each other. Since majority of the cells are of the type LCELLs, we made the virtual platform by stacking together all spaces reserved for LCELLs. In other words, the inter-tile spacing and the space reserved for DFFs, REGs, and BRAMs is removed leaving a contiguous virtual platform which has nearly the same height as the real platform but nearly 25% the width of real platform. The virtual nelist is generated by forcing the cell size of each type of cell to be equal to that of an LCELL. A standard row-based placer can now be exploited to place the virtual netlist onto the virtual platform. In the general case, the space available in the virtual platform may not be sufficient to contain the virtual netlist. In such a case, we expand the virtual platform horizontally until its size is at least 10% larger than the space requirement for virtual netlist to allow sufficient whitespace for the placer.

4.2.2 Transforming Virtual Placement Solution

The results of global placer on virtual platform needs to be mapped back to the real platform. We perform this step by inserting whitespace corresponding to the blockages for LCELLs. This step, which is depicted in Figure 4.3, can cause large increase in wirelength (WL) because several nets which were earlier small would be elongated by the amount of whitespace inserted. Note that in Figure 4.3, we assume only 4 columns of LCELL instead of 24 to avoid cluttering the diagram. To reduce this impact, we perform cut minimization on cells immediately lying on the boundary of the inserted whitespace.

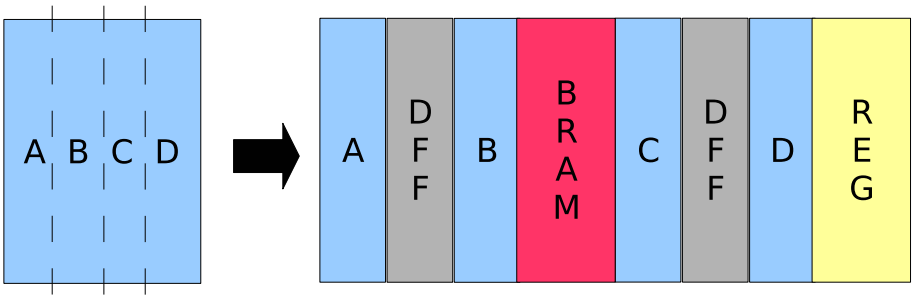

Figure 4.3: Example of shifting of LCELL columns to accommodate other cell types in between. Columns A, B, C and D spread to reform the tile structure

Since the initial placement is performed on a continuous virtual platform, all the cells get placed in a unique tile as a result of mapping the virtual placement solution on to the real platform. In other words, no cell occupies the spaces reserved for DFF, REGs and BRAMs as well as the spacing between individual tiles.

4.2.3 Chip Level Density & Clock Legalization

In the placement obtained from previous step, under the assumption that all cells are of the same type, each tile has density less than one i.e. they can be placed without overlap. However, for all the cells to be placed in their matching cell type, density of each tile for *each* type of cell should be less than one. It is easy to see that if, after all the cells have been assumed to be of type LCELL, there is no density violation, then, there can never be any density violation if some of the cells are turned back into their own type (thus leaving empty space in the space for LCELLs. Due to this reason, we never need to consider density of LCELLs in our approach and only the EDFFs, BRAM and REG file need to be considered.

Satisfying EDFF density: The first step for EDFF density and clock constraint is to determine which set of clocks will be present in each tile under the constraint that no more than N different clocks exist per tile (N=4 in our implementation). Our placer formulates this problem as an ILP as follows:

In a design with T tiles and total of C clocks, let boolean variable E_{ti} denote the existence of clock type i in tile t. The effort of removing k_i cells of clock type i from a tile t is $(1 - E_{ti}) \times k_i$. The contribution of tile t to the cost function is the total effort to clean up the cells of a clock type which cannot be placed in the tile which is $\sum_{i=1}^{C}(1 - E_{ti}) \times k_i$. The complete cost function is simply addition of the above cost function for each tile. To guarantee enough space for all the cells of each clock type, a few more constraints need to be

put.

$$\text{Minimize} \qquad \sum_{t=1}^{T}\left(\sum_{i=1}^{C}(1-E_{ti})\times k\right)$$

$$\text{Subject to :}$$

$$\sum_{i=1}^{C} E_{ti} \;\leq\; N \qquad \forall t \;\in (1,T)$$

$$\sum_{t=1}^{C} E_{ti} \;\geq\; Min_i \;\; \forall i \;\in (1,C)$$

$$E_{ti} \qquad \in \;\; 0,1$$

where Min_i is the minimum number of tiles required for the DFF with clock i for the legalization to be possible. The above ILP formulation has O(T×C) boolean variables, O(T) constraints for guaranteeing no more than N clocks in each tiles, and O(TC) constraints to guarantee enough space for all cells. Considering that T is in hundreds and N is in tens, the above ILP formulation can be solved within seconds.

Once the clock assignment is fixed, we formulate the problem of determining how many DFFs to move among various tiles as a min-cost network flow problem with capacity constraints. The network flow is constructed by connecting a tile congested w.r.t. DFFs with the four tiles around it. The maximum capacity of any edge is taken as the minimum of the number of DFFs that need to exit a congested tile and the number of DFFs which can move into neighboring tile without congesting it. In our implementation, the cost of moving a cell out of its current tile is 1. Thus the network flow solution minimizes the *number of DFF* that need to move for satisfying density constraints. It is possible to extend our work by modeling the cost as change in

WL due to moving a DFF which corresponds to a solving density constraints with least movement. At this stage, each tile satisfies density constraints w.r.t. DFFs.

Satisfying REG Density: Since each tile can have at most 4 REGs, and the number of REG cells to be placed are usually very small (several hundreds), it is possible to solve their placement globally. For T tiles, there are 4T valid places for REGs. We cast this selection as an assignment problem which can be solved efficiently using munkres algorithm [63] whose efficient implementation is O(n^3) complexity. The cost of placing a REG into a valid position is modeled as the wirelength(WL) of all the nets incident to the REG cell in that position. The solution to the assignment problem not only identifies which REG goes to which tile, but also fixes its position to a valid location in the tile. Even for the biggest benchmark and platform, this step takes less than 30 seconds CPU time.

Satisfying BRAM Density: The case for BRAM is exactly the same as that of REGs and can be solved by assignment problem readily. Notice that owing to very small number of BRAMs and only one space per tile, the size of assignment problem for BRAM is an order of magnitude smaller than that for REGs and is solved within 10 seconds for biggest benchmark.

4.2.4 Tile Level Site Legalization

Once all of the above steps are run, density of each tile for each type of cell is under control. As pointed out earlier, the BRAMs and REGs are already

placed at their best legal location by virtue of the solving assignment problem which maps them to legal locations. However, the LCELLs and EDFFs need to be put in their respective columns (see Figure 4.1). There are two main steps to perform this function:

Column's clock type determination: In the first step for tile level site legalization, we determine which *column* should be reserved for which clock (recall that each column can have cells of only one type of clocks). Our experiments show that this step is very critical since if the column reservation is not done smartly, wirelength can degrade significantly when moving the EDFFs to column which can accommodate its clock type. We solve this problem using ILP on the same lines as solving the problem of deciding which clocks can come in which tile (see Section 4.2.3). Each column c is assigned as many binary variables as their are clock types. The cost function is the sum of efforts required to clean up a column by removing the EDFFs whose clock type is not the same as that returned by ILP solution. A column which initially does not have any EDFF in it is assigned a clock number -1 indicating that it can be reserved at a later time while wirelength minimization. Consider a case where upto M EDFFs can be accommodated in a column and let each tile have P EDFF columns and currently C clocks in it with N_i (i $\in$ [1, C]) EDFFs for each of these clock types. Binary variable p_i, when true, expresses that column p is reserved for clock i. Let the number of cells before EDFF site legalization in the column p of clock type i be given by n_{pi}.

The complete ILP can be written as:

$$\begin{aligned}
\text{Minimize} \quad & \sum_{p=1}^{P}\left(\sum_{i=1}^{C} p_i \times \left(\sum_{j=1,j\neq i}^{C} n_{pj}\right)\right) \\
\text{Subject To :} \quad & \sum_{i=1}^{C} p_i \;\leq\; 1 \qquad\qquad \forall p \in [1,P] \\
& \sum_{p=1}^{P} p_i \;\geq\; \lceil n_{pi}/M \rceil \quad \forall i \in [1,C] \\
& p_i \quad\in\; 0,1
\end{aligned}$$

The ILP above has O(PC) binary variables and O(P + C) constraints. The first set of constraints make sure only one clock can occupy a column. The second set of constraints guarantee enough columns reserved within the tile for each clock type so that its cells can be placed. The values of C and P is determined by the architecture of structured ASIC and is generally a small constant number (e.g. $C = 4$ and $P = 24$ in our benchmarks). In our experiments we observed that the above ILP takes less than 0.2 seconds to solve.

Site Legality of LCELLs and EDFFs: With the assignment of all columns to a clock type or unreserved type, we iterate over the LCELLs and EDFFs to legalize them. These cells are sorted in non-decreasing horizontal coordinate and are assigned to their closest unoccupied and clock compatible (in case of EDFFs) possible location. A procedure *rotateAndPlace* has been implemented which takes as input a given point and a cell and rotates in circle with increasing radius around the point until the given cell can be placed legally depending on the type of the cell.

4.2.5 Wirelength Recovery

The chip and tile level density legalization outlined in the Section 4.2.3 and Section 4.2.4 produce placement results which are completely legal w.r.t. clock, overlap and site constraints. However, due to movement of several cells away from the initial relative order suggested by the placer results on virtual platform, significant increase in the wirelength occurs. Our wirelength optimization procedure depicted in Algorithm 5 recovers this increase in wirelength. Our philosophy during wirelength minimization is to never break the clock, site or overlap legality. Algorithm 5 shows the three main phases of our wirelength recovery. In the first phase, large distance inter-tile movement of cells takes place. Using the median idea [49], the best bounding box (BB) of each cell is computed. The cells are then sorted in the non-increasing order of their distance from their best BB and processed in that order. For the cell being currently optimized, a procedure returns the list of all the tiles which have non-empty geometrical intersection with its best BB. This list is iterated while trying to place the cell near the center of the intersection of best BB and the tile currently being tried using routine *rotateAndPlace*. After this stage, all inter-tile movements have been done and we focus on intra tile wirelength minimization. The second phase improves adjacent columns in the tile to identify any horizontal movement of cells. This is achieved by looking at each consecutive pair of columns such that both are of the same type (EDFF or LCELL). Further, if both the columns are EDFF columns, their clock type should be same. For such a pair of row, horizontal swapping of a cell in first

with the neighboring cell or space in the second column is tried. In the third phase, we look at three consecutive cells/spaces in each column starting from lowest vertical position and enumerate all six possible combinations and pick the best. In all the three phases, we check to see if a movement has caused WL increase and if so, we revert back the change. Our third phase is similar to the method outlined in [78] with the major difference being that we consider white-space also during explicit combination enumeration while FastPlace does not.

4.2.6 Experimental Setup and Results

All the above algorithms are implemented in C++ in our placer tool RegPlace (REGular PLACEr). All our experiments were run on a dual-core 3.3GHz 64-bit AMD linux workstation with 4GB RAM and 8GB swap. We used GLPK [5] as the ILP and network flow solver. The benchmarks were provided by eASIC as part of the placement challenge. The initial placement on the virtual platform were generated using two global placers: mPL [34] and CAPO [82].

Table 4.2: Benchmark Characteristics

Bench	#LCELL	#DFF	#REG	#BRAM	#CLK
easic1	832,824	87,052	110	172	18
easic2	812,200	45,478	175	686	7
easic3	961,063	52,780	192	0	3
easic4	102,038	23,330	0	44	7
easic5	913,853	84,505	145	262	26

Algorithm 5 Wirelength Minimization

Ensure: Placement Is Legal

{ **Large Distance Movement** of Cells}

while WL Improvement $\geq \delta$ **do**
  Compute Best BB b_c for each cell c
  Sort cells according to distance from b_c
  for all cell c in sorted list **do**
    List lst = All tiles intersecting b_c
    for all tile t in list lst **do**
      rotateAndPlace(c, center of t's intersection with b_c)
      if could place in t **then**
        if WL Improved **then**
          break
        else
          continue

{ **Intra Tile Inter Column** WL Improvement}

for all tile t in platform **do**
  while WL Improvement $\geq \delta$ **do**
    for all adjacent columns $p1$ & $p2$ **do**
      if $p1$ and $p2$ are not same type or $p1$ & $p2$ have different clocks **then**
        continue
      for all vertical positions in columns **do**
        Swap cell-cell/cell-space pair adjacent in $p1, p2$
        if WL NOT Improved **then**
          Swap back to original position

{ **Intra Tile Intra Column** WL Improvement}

for all Tile t in platform **do**
  while WL Improvement $\geq \delta$ **do**
    for all Column p in t **do**
      Sort cells and spaces in p by vertical coordinate
      Try all 6 combination of 3 consecutive cells/space
      Pick the best configuration

4.2.7 Advantage of VP Generation

To prove the efficacy of generation of virtual platform (VP) instead of solving the placement problem by blockages, we did the following experiment:

Existing row-based placers were run on two benchmarks which were identical to each other except that in one case the placer was run on virtual platform and then mapped back to real platform whereas in the second case, the placer was run by considering blockages. In both the cases, all the cell types were converted to LCELLS, and only the space for LCELLS were available to the placers. Table 4.3 shows the WL results and time taken to complete the placement for several benchmarks.

Table 4.3: Advantage of using Virtual Platform (VP) to generate placement solution compared to placement with blockages (WB) when using CAPO. Wirelength (WL) are in millions of units. All CPU time are in minutes.

	WL			CPU		
	VP	WB	Δ	VP	WB	Δ
easic1	11.1	11.1	0%	230	464	2.0X
easic2	21.3	21.2	-0.47%	231	476	2.1X
easic3	17.1	17.3	1.15%	374	639	1.7X
easic4	2.0	2.1	5%	29	133	4.5X
easic5	14.55	15.2	4.2%	247	516	2.1X
Average			2.6%			2.5X

Table 4.3 shows the advantage of using VP to migrate the discrete placement region into a contiguous region compared to the approach of specifying unplacable area as blockages. Overall, by using VP method, the wirelength and runtime improves by 2.6% and 2.5X respectively. On further analysis, we found that most of the extra time is spent by CAPO in overlap removal due to lots of overlaps of cells with the blockages.

While mapping the solution of placement on VP back to real platform, we re-introduce all the blockages. As pointed out in Section 4.2.2, the wire-

length increase due to this step can be reduced by reducing the cut-size by swapping cells lying immediately on the both sides of the introduced blockage. To quantify this benefit, we performed cut-minimization during mapping step on our placement results of Table 4.3. Only the cells upto two unit distance away on both sides of the blockage re-insertion point were considered during cut minimization to reduce the performance overhead. On an average, we are able to further reduce the WL by 2% over the numbers in previous table.

Table 4.4: Comparison of our placer with other contestants. CPU numbers for Team 2 unavailable.

Bench	Team1		Team2		RegPlacer (M)		RegPlacer (C)	
	WL	CPU	WL	CPU	WL	CPU	WL	CPU
easic1	16.9M	3499	10.6M	N.A.	11.3M	13849	12.8M	14538
easic2	32.3M	2444	25.8M	N.A.	23.4M	9529	25.1M	9720
easic3	20.9M	2907	20.5M	N.A.	18.8M	6627	19.6M	7045
easic4	2.3M	216	1.8M	N.A.	2.0M	695	2.1M	735
easic5	20.4M	3055	14.4M	N.A.	13.9M	10197	16.6M	10479
Total	92.8M	12122	73.1M	N.A.	69.4M	40897	76.2	42517

A preliminary version of RegPlacer competed and won the eASIC placement contest [4]. In its current form, we have further improved the wirelength and runtime of RegPlacer by 9% and 10% respectively. Table 4.4 shows the comparative data for our placer vs. other participants Team1 The WL numbers for Team1, Team2 are as reported in the contest results. However, since we have upgraded our placer, the runtime shown for Team1 and Team2 teams have been scaled by the proportion of runtime our initial version of placer takes on our workstation compared to the contest benchmarking workstations. To understand the impact of using different global placers to generate an initial

solution on the virtual platform, we repeated our placer flow with two state-of-the-art placer: mPL and CAPO. The initials (M, C) in last four column names depict use of the above placers respectively.

From Table 4.4, several observations can be made. Our wirelength results improve significantly (10%) when the initial placement on virtual platform is generated using mPL rather than CAPO. Similarly, the runtime improves on using mPL. Compared to other teams, RegPlacer beats Team1 in all the benchmarks irrespective of the initial placer solution being generated with CAPO or mPL. The wirelength reduction w.r.t. Team1 is as much as 33%. However, Team1 does have the advantage of being very fast (3.3X faster than RegPlace). RegPlacer when used with mPL for initial placement, is better than Team2 in 3 out of 5 benchmarks as well as the total wirelength. Due to absence of runtime information of Team2, we cannot compare the runtime. The reduction in total wirelength compared to Team2 is 5%. These results show that RegPlace is a high quality solution for structured ASIC placement problem. Figure 4.4 shows the output of our placer for the benchmark easic2 where each cell type has been plotted with a different color.

4.2.8 Discussions

In this section, we proposed a new flow for efficient solution of placement problem for structure ASICs. The key novelty of this work are: a) concept of virtual platform for obtaining better initial placement solution, b) network-flow based inter-tile density correction, c) very fast ILP based clock

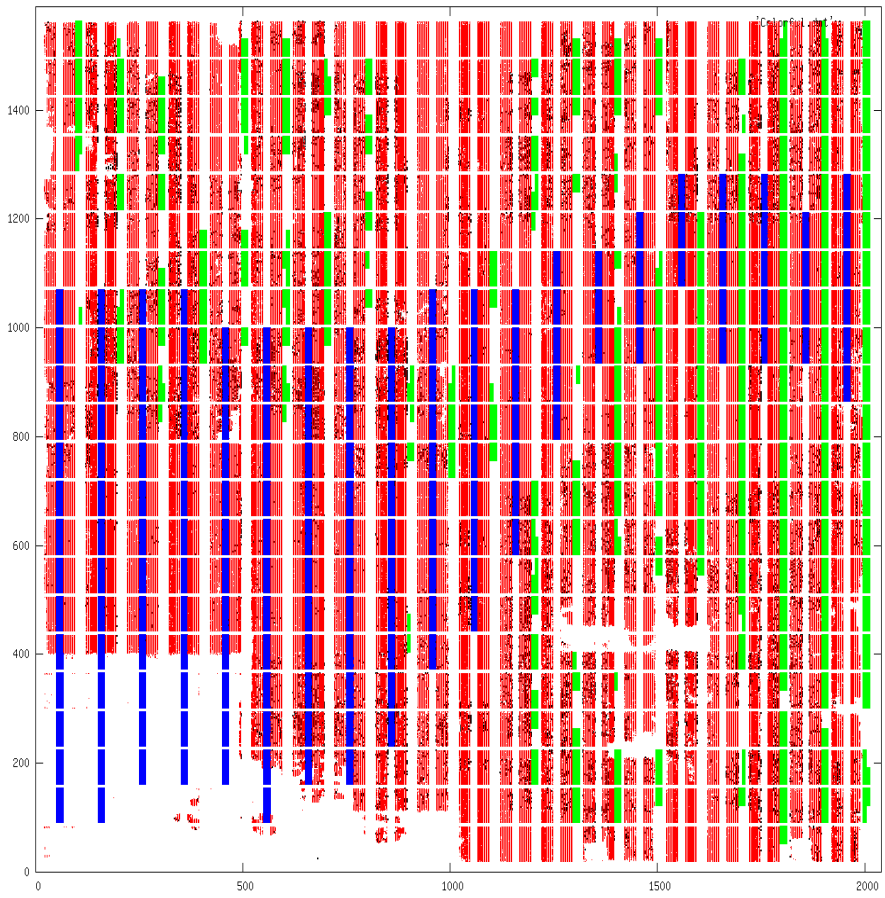

Figure 4.4: Final layout of benchmark easic2. Cells are shown as LCELL(red), DFF(black), REG(green), and BRAM(blue).

constraint correction at both chip and tile level. Powered by these techniques, our placer won the eASIC placement challenge. Experimental results show that we obtain as much as 33% less wirelength than the competing teams with only 3.3X increase in CPU time.

4.3 Low Power Placement

Leakage Power Model: The leakage power of a structured ASIC platform is roughly proportional to the number of devices in it. Even if no cell (in the RTL) is mapped on to a device on the platform, the device will still consume leakage power unless disconnected from the power supply. Existing structured ASICs already allow power down (i.e. disconnect from power grid) of unused devices on the platform using single-via power programming. We assume that power-down can be performed at two levels of granularity: a) An unused tile can be completely powered down reducing its leakage (and dynamic) power to zero, and b) Within a tile, an unused column can be powered down if no functioning cells are placed in it, reducing its power to zero. The leakage power savings achieved by power down of a complete tile is equivalent to that achieved by individual power down of each column in the tile individually. If the leakage power of one column is P_{col}, then the leakage power model of the structured ASIC platform is simply the sum of leakage power of all the columns that are not power down.

$$P_{\text{leak}} = P_{\text{col}} \times \sum_{t \in \text{tiles}} \left(\sum_{c \in \text{cols(t)}} E_c \right) \tag{4.1}$$

where cols(t) denotes the columns in tile t and E_c is a binary variable whose value is 1 if the column c is powered on and 0 otherwise. From Eqn 4.1, it is clear that we should try to minimize the number of columns powered up to reduce leakage power.

Clock Power Model: Unlike leakage power, clock power is largely proportional to the clock switched capacitance. Due to stringent slew and skew constraints on the clock signal, clock is distributed across the platform using hierarchy of high drive power hungry buffers. To reduce clock power, the aim should be to reduce switched capacitance of the clock network. To understand the clock power model, we must first understand clock distribution architecture. We assumed a simplistic clock distribution network as depicted in Figure 4.5 described using 2x2 tiles to reduce clutter. Figure 4.5-a depicts the platform level clock distribution whereas Figure 4.5-b depicts the tile level clock distribution. The architecture we assume is based on the recent works on clock distribution on FPGAs [97][96].

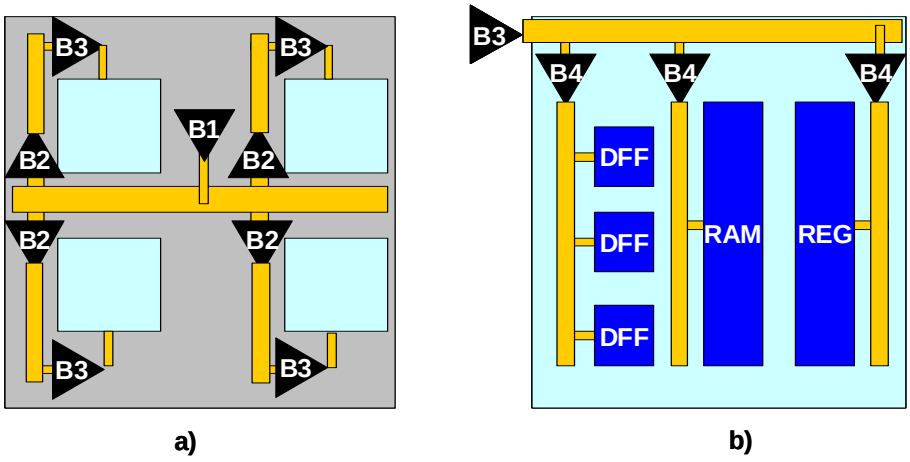

Figure 4.5: Clock distribution: a) Platform b) Tile level

Here, we describe the clock distribution architecture of a single clock which is duplicated to distribute multiple clocks. In Figure 4.5-a, level 1 buffer

B1 inserts the clock signal at the center of the chip that drives a horizontal spine. Each level 2 buffer **B2** distributes the clock signal to one vertical spine. Similarly, a level 3 buffers **B3** drives the clock signal into each tile. Inside a tile, as shown in Figure 4.5-b, Level 4 buffers **B4** drive the `DFF` columns (only 1 shown) as well as `REG` and `RAM` cells. Owing to the larger number of sequential elements in `RAM` and `REG` cells, multiple **B4** buffers are required to drive them (only 1 shown). We assume that each **B4** buffer can drive only as many sequential elements as are in a `DFF` column. To distribute multiple clocks, the **B1** and **B2** clock buffers can be replicated for each clock signal. Due to routing constraints, typically only a subset of platform level clocks can be routed into each tile. Each **B3** and **B4** clock buffers is replicated as many times as the size of subset of clocks that can be routed in each tile. We assume that power-down of clock signal can be performed at three levels of granularity: a) Clock signal to an unused vertical spine can be disconnected if all of the tiles driven by this spine are unused. b) An unused tile can be disconnected if all cells in it are unused, and c) within the tile, the clock driving each `DFF` column, `REG`, `RAM` cell can be disconnected.

For platform with WxH tiles (instead of 2x2 in Figure 4.5), let C_G clocks at platform level and C_T clocks at tile level be supported. Further, let the number of equivalent (after scaling for larger sizes of `REG` and `RAM`) number of `DFF` columns be S each containing N sequential elements. Under these assumptions, Table 4.5, shows the constituents of the clock network. Columns 2 contains the number of buffer needed whereas column 3 shows the

capacitive load on each of these buffers. The last column shows the total switched capacitance seen by the corresponding level of clock buffers when the input capacitance of an entity X is denoted by L_X.

Table 4.5: Clock Tree Distribution Characteristics

Buffer Type	Number Instance	Each Load	Total Load
B1	C_G	2 W L_{B2}	2 W C_G L_{B2}
B2	C_G 2 W	0.5 H L_{B3}	2 W H C_G L_{B3}
B3	C_T W H	S L_{B4}	W H S C_T L_{B4}
B4	C_T W H S	N L_{DFF}	W H S N C_T L_{DFF}

Using the clock network architecture and Table 4.5, the total clock power can be written as

$$P_{\text{clk}} = \sum_{i=0}^{C_G} \left(L_{B1} + \sum_{j=0}^{2W} \left(E_{ij} L_{B2} + \sum_{k=0}^{0.5H} (E_{ij} E_{jk} L_{B3}) \right) \right) + \sum_{t \in \text{tiles}} \left(\sum_{m=0}^{C_T} \left(\sum_{n=0}^{S} (E_{mn} L_{B4} + E_{mn} L_{DFF}) \right) \right) \tag{4.2}$$

where all E_{xy} are boolean variables. E_{ij} is 1 if clock i drives spine j, E_{jk} is 1 if spine j drives tile k on that spine, E_{mn} is 1 if clock m in the tile drives the column n. The first term in Equation 4.2 is the total switched capacitance of C_G global clocks and the second term is the total switched capacitance within a tile, summed over all the tiles. Selective power down of unused components (spine, tile, column) modulate the values of these binary variable thus changing the clock power.

We now introduce the problem being tackled in this work.

Given a structured ASIC platform and a design, perform cell placement to minimize the clock and leakage power consumption by maximizing the components that can be powered down. Power reduction should not degrade the wirelength by more than a given budget.

4.3.1 Overall Flow

Figure 4.6, shows our complete flow. The *rectangular* blocks represent our new contributions whereas the oval blocks represent the steps in `RegPlace` [30], an existing open source placer for structured ASICs. After describing the overall flow in this section, each rectangular block will be explained in more details in next few subsections.

Given the input netlist and the platform, we minimize the vertical spines and tiles that must remain powered up by generating new constraints for global placer. This step works under a given wirelength increase budget. Global placement is then executed on the design followed by clock legalization. In the next step, we minimize the number of columns that must stay powered on by avoiding scenarios where the number of cells in a tile is slightly more than integral multiple of number of cells that can be accommodated in a column, thereby requiring another column. Finally, we cluster and assign the cells into as few columns as possible while minimizing wirelength change.

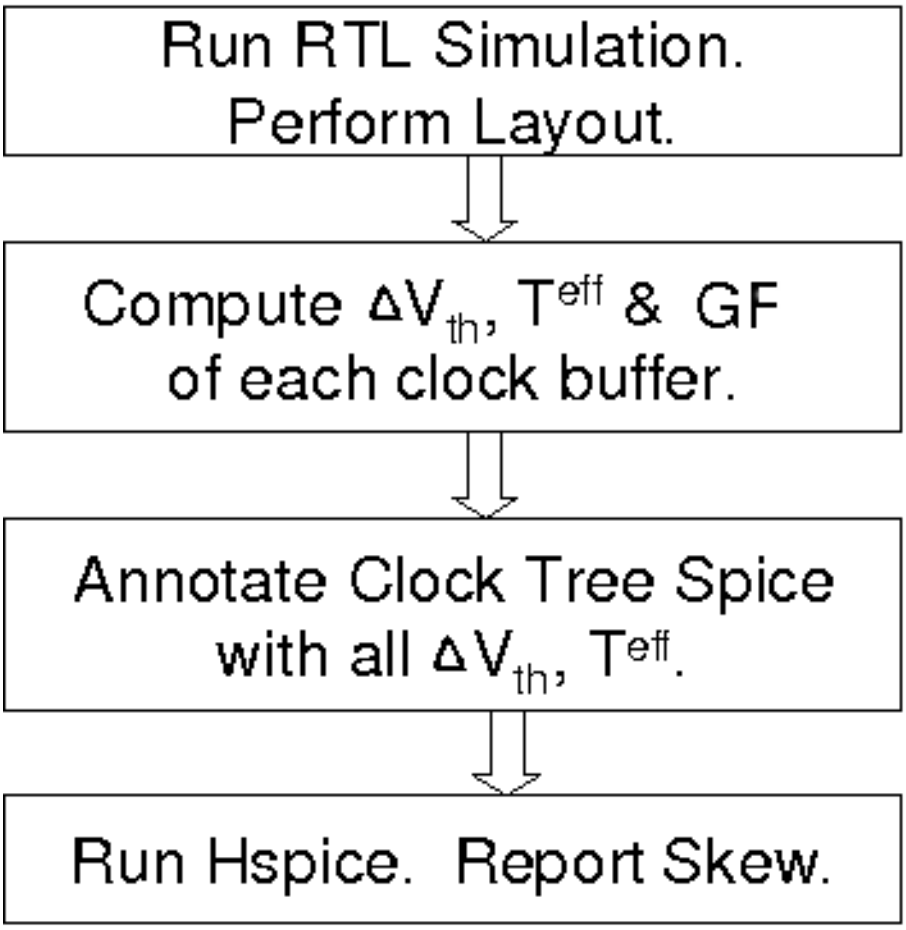

Figure 4.6: Our placement flow. Blue rectangles are our new contributions. Grey ovals are from [30].

4.3.2 Spine and Tile Reduction

To mark a vertical spine as unused, we must ensure that *all* the tiles driven by the spine are marked as unused. To mark a tile as unused, we must make sure that no cell is placed in it. If the design to be placed is much smaller than the physical platform, naturally there would be many such tiles.

However, our aim is to maximize such unused tiles by identifying those which would have very low number of cells in them and evicting these cells without much wirelength penalty.

Algorithm 6 depicts our overall spines and tile reduction strategy. We performed rough placement to generate a coarse density map ensuring that the cut-line during partitioning coincide with the tile boundaries (Line 1). The wirelength increase budget is set to $k\%$ of the original wirelength (Line 2). For each horizontal trunk in the platform, we go over all the vertical spines connected to it. These spines are sorted in non-decreasing order of the number of cells in all the tiles driven by the corresponding vertical spine (Line 4-5). Then, the sorted set of spines is iterated over while trying to evict the cells in it using procedure `EvictSpine`. The scheme for reducing the number of tiles is also similar (Line 8-10) wherein the procedure `EvictTile` is executed for each tile in sorted order. We now describe Algorithm 7 and Algorithm 8 which implement procedure `EvictSpine` and `EvictTile` respectively.

Algorithm 7 which tries to evict a given spine works as follows. For a target current spine being evicted, the list of all tiles in the spines to the left and right of it is obtained (Lines 1-2). Next we write a network flow formulation where all the cells from each tile belonging to the current spine are moved out of the spine. For each tile in the spine, the possible location of interest are all the tile adjacent to it except itself (Line 3-5). We limit the cells to move to only adjacent tiles to reduce wirelength penalty. The objective of the network flow is to minimize the estimated wirelength increase due to

Algorithm 6 Vertical Spine and Tile Minimization

```
1: Perform Rough Placement
2: BUDGET = k % of wirelength of design {Minimize spine utilization in
   next block}
3: for all Horizontal Spine hs do
4:    CONN = All vertical spines connected to hs
5:    Sort CONN by number of cells in it
6:    for all Vertical Spine vs connected to CONN do
7:       EvictSpine(vs, BUDGET)
   {Minimize tile utilization in next block}
8: T = All tiles in platform
9: Sort TILES by number of cells in it
10: for all Tile t in TILES do
11:    EvictTile(vs, BUDGET)
```

Algorithm 7 EvictSpine(Spine S, BUDGET)

```
1: SL(SR) = Spine on left(right) of S
2: REST = Tiles in SL ∪ SR
3: for all Tile t in S do
4:    NGHBR = All tiles in REST <= 1 tile away from t
5:    Build flow between t and NGHBR
6: Add Constraint for WL Increase < BUDGET
7: if Flow Possible then
8:    Move cells from tiles in S to SL or SR
```

the movement. An additional constraint is added to bound the maximum acceptable wirelength increase (Line 6). If the network flow is possible, the spine S can be evicted and its cells are moved out.

Algorithm 7 can only evict one whole spine at a time. However, there can be many scenarios where a few (but not all) tiles of a spine have small number of cells in it and can be evicted. For this, Algorithm 8 works on each tile in the design. For each tile, we find the average x and y coordinate (XAvg

Algorithm 8 EvictTile (tile `T`, `BUDGET`)

```
1: Get xAvg = average X coordinate of cells in T
2: Get yAvg = average Y coordinate of cells in T
3: Compute Up/Dn/Lf/RtConn for T
4: NGHBR = All neighbors of tiles T
5: for all tiles n in NGHBR do
6:    if n is an empty tile, continue;
7:    moveX = n's center X coordinate - xAvg
8:    moveY = n's center Y coordinate - yAvg
9:    WLINCR = moveX × (LfConn - RtConn)
10:   WLINCR += moveY × (DnConn - UpConn)
11:   Add flow from T to n with cost WLINCR
12: Add Constraint for WL Increase < BUDGET
13: if Flow Possible then
14:   Evict T and move cells
```

and `YAvg`) of the cells in it (Line 1-2). We also compute four numbers (RtConn, UpConn, LfConn, DnConn) which represent the number of nets incident on any cell in the tile whose bounding box is strictly to the right, above, on left, or below this tile (Line 3). The intuition is that if all the cells of this tile are moved to another tile, the increase in wirelength by this move can be approximated by combination of these numbers. Next, we iterate over all eight physically contiguous tiles around the target tile (Line 5) which are the possible destinations while computing the estimated wirelength increase (Line 7-10). A network flow problem is built and if there is a legal solution to it, the tile `T` can be evicted (Line 13-14).

After the termination of Algorithm 6, we obtain the largest set of complete spines and tiles which can be evicted under given wirelength increase budget. Based on this information, we transform the placement problem by

adding fixed blockages covering the unused tiles. This transformed placement problem becomes the input to `RegPlace` global placer.

4.3.3 Column Minimization

After the global placement and clock legalization has been performed, combinational (i.e. `LOGC`) and sequential cells belonging to multiple clocks could be distributed in each tile. To exploit this flexibility, we apply Algorithm 9 to a rectangular peephole M tiles wide and N tiles high. Once the number of powered up columns in the peephole is reduced, the peephole is moved to the next adjacent position and Algorithm 9 is rerun. If the whole platform consists tiles that is W wide and H high, Algorithm 9 is executed W-M $\times$ H-N times. Now we describe the steps involved in one invocation: We add a dummy clock called "NIL" which connects to all combinational cells (i.e. `LOGC` cells) (Line 3). The optimization is performed for one clock type at a time (Line 4). The maximum number of cells that can be accommodated in a column is given as N (Line 2). We prune the cases for which the number of columns occupied is already optimal (Line 6-9). For each tile in the peephole, we compute the minimum count of cells (`Extra`) that, when removed, leave an integral multiple of N cells (Line 10-11). All the tiles in the peephole are sorted according to non-decreasing `Extra` count (Line 12). For each tile, every tile except itself is a prospective tile in which the `Extra` cells can be moved (Line 14). To reduce the impact on wirelength increase, we sort the prospective tiles according to distance from the current tile (Line 15) and try

Algorithm 9 Column Count Reduction

```
1:  T = All tiles in current peephole
2:  N = Maximum cells in a column
3:  C = All clocks in T ∪ "NIL"
    {"NIL" Clock connects to combinational cell i.e. LOGCs}
    {Let n_ci = cells of clock c in tile i}
4:  for all Clock c ∈ C do
5:     MOVES = φ
6:     USED = Σ_{i∈T} ceil(n_ci/N)
7:     REQD = ceil(Σ_{i∈T} n_ci)/N
8:     if USED == REQD then
9:        continue; {Column count already optimized}
10:    for all Tile t ∈ T do
11:       t.Extra = n_ct - N × floor(n_ct/N)
12:    Sort T in non increasing order of Extra(t)
13:    for all Tile t ∈ sorted T do
14:       OTHER(t) = T - {t}. Sort w.r.t. distance from t.
15:       for all Tile o ∈ sorted OTHER do
16:          trans = min(t.Extra, N-o.Extra)
17:          MOVES = MOVES ∪ {trans from t to o}
18:          if t.Extra == 0 then
19:             break
20:    Implement all MOVES stored
```

pushing the `Extra` cells of current tile into them as long as this move does not create new columns in the prospective tile (Line 17-18). Once all the moves to reduce the column count for a clock are recorded, we actually implement them by moving the cells among the tiles. When moving a cell from tile A to tile B, its new location in tile B is chosen as the point closest to the original location in tile A.

4.3.4 Column Assignment

After the termination of Algorithm 9, the decision regarding clustering of these cells into columns and how the columns for the different clocks are relatively placed, still needs to be made. For this, we use Algorithm 10 which runs on one tile at a time and packs the cells into individual columns. Algorithm 10 first performs greedy clustering of cells of same clock type into

Algorithm 10 WL Aware Column Assignment

```
1: COLS = Columns in the tile
2: N = Capacity of each column
3: BINS = φ
4: C = All clocks in tile ∪ "NIL"
   {"NIL" Clock connects to combinational cell i.e. LOGICs}
5: for all Clock c ∈ C do
6:    CLSTR = φ
7:    L = cells of clock c from left to right
8:    for all Cell i ∈ L do
9:       CLSTR = CLSTR ∪ i
10:      if size of CLSTR == N then
11:         BINS = BINS ∪ CLSTR
12:         CLSTR = φ
13: Assign BINS to COLS [assignment problem].
    Cost of assignment = incident wirelength.
14: Swap cells between columns of same clock to reduce WL
```

bins of size N (=capacity of each column) starting the sweep from the left edge of the tile (Line 9-12). All the cells belonging to one bin form one cluster and placed in one column. If the number of bins used for packing is b and the number of columns available in the tile is n, mapping the bins to n columns can be cast as an *Assignment Problem* and solved efficiently using Munkres

algorithm (Line 13). Finally, to repair wirelength degradation due to sub-optimal clustering of cells into columns, cells are swapped between columns of same clock type if it reduces wirelength.

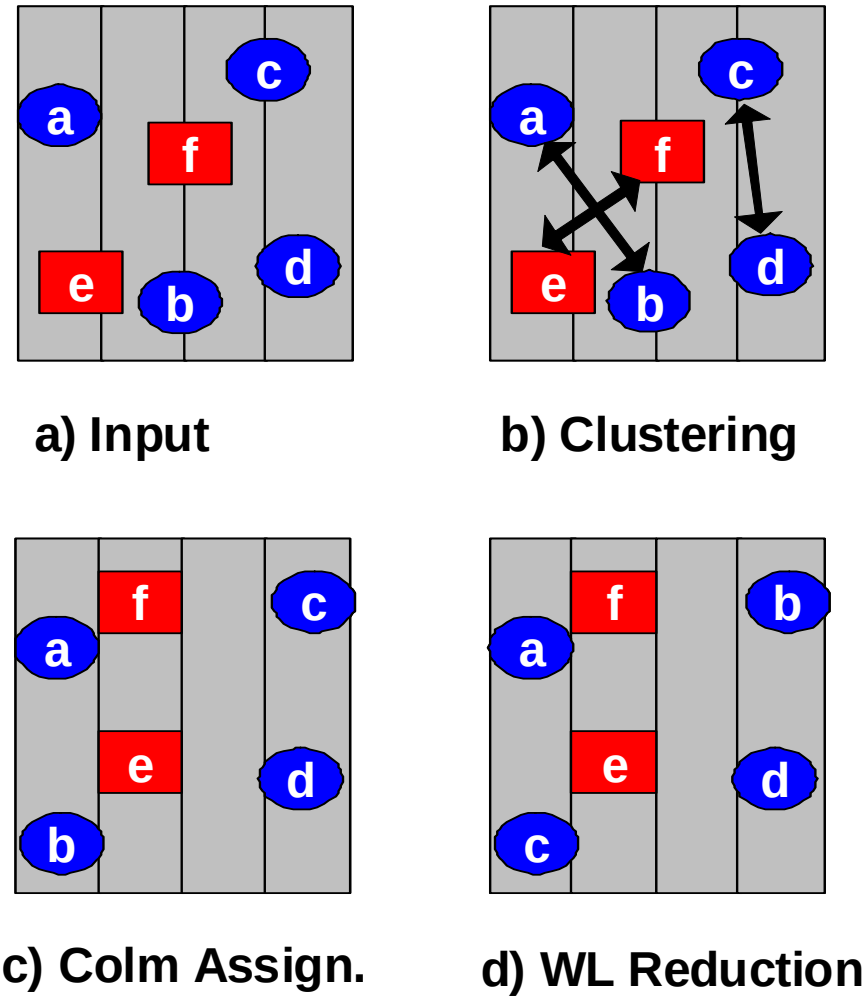

Figure 4.7: Column assignment: a) Input. b) Cells clustered based on original location. c) Columns assigned to each cluster. d) After WL reduction by swapping

Figure 4.7 shows an example of column packing in action. For this example, each column can have only 2 cells. Cells driven by the same clock type have the same shape. The cells are divided in three bins as shown in Figure 4.7-b. These bins are assigned to individual columns as shown in Figure 4.7-c. Finally, cell swapping may lead to swapping of cells b and c to reduce wirelength.

4.3.5 Experimental Setup

We implemented our low power placement flow `PASAP` in the C++ language inside the open source structured ASIC placement tool `RegPlace` [30]. All experiments were run on a 16 core, 1.6 GHz, Linux workstation. We used GLPK [5] as the integer linear programming (ILP) and network flow solver and a custom implementation of Munkres [63] algorithm for assignment problem. The wirelength increase budget in Algorithm 6 was set at 15%.

Platform: Two different platforms were generated by replicating the *tile* described in Section 4.1. The details of these platforms are in Table 4.6. Columns 2 and 3 show the number of times the tile is replicated in horizontal and vertical direction respectively. Rest of the columns indicate the maximum number of `LOGC`, `DFF`, `REG`, `RAM` that can be accommodated in the platform.

Table 4.6: Platform Characteristics

Plat Name	Rep X	Rep Y	Max `LOGC`	Max `DFF`	Max `REG`	Max `RAM`	Max CLK
A	20	22	1M	675k	1760	440	32
B	22	24	1.2M	811k	2112	528	32

Designs: The benchmarks used were original released by `eASIC` for the placement contest. Table 4.7 shows the breakup of the design in terms of constituent cells. The total number of cells ranges between 125K to over 1 million. Column 6 shows the number of global clock signals used.

Table 4.7: Benchmark Characteristics

Name	LOGC	DFF	REG	RAM	CLK
easic1	832,824	87,052	110	172	18
easic2	812,200	45,478	175	686	7
easic3	961,063	52,780	192	0	3
easic4	102,038	23,330	0	44	7
easic5	913,853	84,505	145	262	26

4.3.6 Results

Table 4.8 describes our experimental results comparing the placement generated by `RegPlace` to `PASAP` placement. The benchmark name in the first column also shows the platform on which the benchmark was placed. For example `easic4B` identifies that it is the benchmark easic4 placed on platform B. The metrics being compared for the two placement are : a) unused spines, b) unused tiles, c) unused columns, d) wirelength and e) total CPU runtime. For each of these metrics, three sub-columns are shown in Table 4.8. These sub-columns depict the value obtained using `RegPlace` (`RP` in table), value obtained using PASAP (`PA` in table) and the difference between these two solutions respectively.

The efficacy of `PASAP` to increase the number of unused spines, tiles and columns is evident from Table 4.8. Due to the different benchmark sizes and

utilization ratio, the numbers vary widely among the different circuits. Therefore, in our discussion, we will primarily focus on the improvement/penalty summed over all the benchmarks. First of all, we note that over all the benchmarks, nearly 58% extra clock spines can be turned off using our method. Turning off spine directly corresponds to lower clock capacitance. Benchmarks easic1A and easic3B has such high utilization ratio that both `RegPlace` and `PASAP` cannot find even one unused spine. Similarly, the number of unused tiles can be increased by around 29% thus giving significant enhancement in the power saving ability. As for the number of unused columns, their number improve by approximately 42% using `PASAP` as compared to `RegPlace`. The very low improvement in the number of column used for benchmark easic4A and easic4B is due to its very small size which implies even by using `RegPlace`, most of the columns are unused. The above improvement in increasing the count of unused spine, tiles and column comes at a cost as seen in the wirelength and runtime penalty. The wirelength of the designs increase by approximately 17%. Due to the runtime associated with a rough placement generation and aggressive spine/tile/column reduction, `PASAP` takes approximately 30% more runtime that `RegPlace`. Though runtime degradation is not insignificant, we believe it is a fair trade-off for the kind of power savings possible with `PASAP`.

To compare the power reduction, the change in placement metrics from Table 4.8 must be converted to clock and leakage power. Using the relation derived in Eqn 4.1, the leakage power saving achieved by performing placement using `PASAP` as compared to `RegPlace` is plotted in Figure 4.8. Each column

in Figure 4.8 corresponds to one benchmark with the exception of last column which shows the average savings. PASAP reduces the leakage power in the range of 7% to 40% with an average value of 17%.

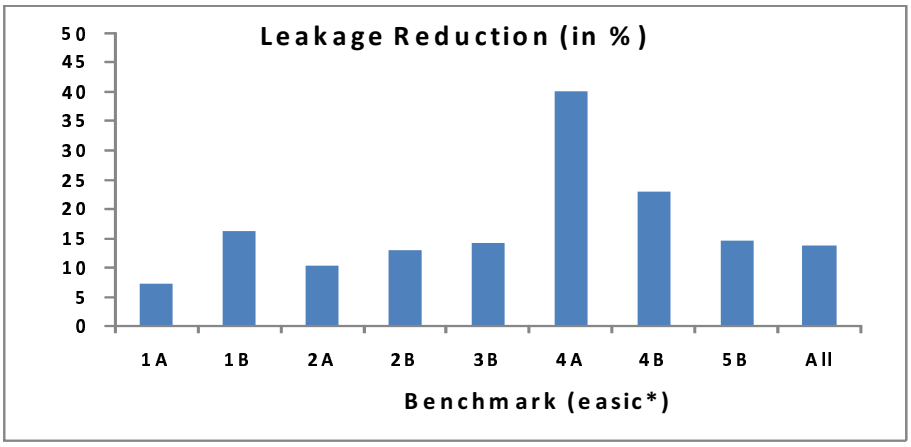

Figure 4.8: Leakage power savings for placement generated by PASAP compared to RegPlace.

Next we compare the clock power reduction due to PASAP. By aggressive reduction in the number of spines, tiles and columns in which the clock signal needs to be distributed, PASAP can significantly minimize the clock switched capacitance. Based on the clock power expression derived in Eqn 4.2, we plotted the clock power reduction achieved by PASAP for different benchmarks in Figure 4.9. PASAP reduces the clock power of different benchmark circuits in the range of 20% to 60% with an average value of 32%. The clock power was computed assuming the frequencies of all the clock signal in the structured ASIC is the same.

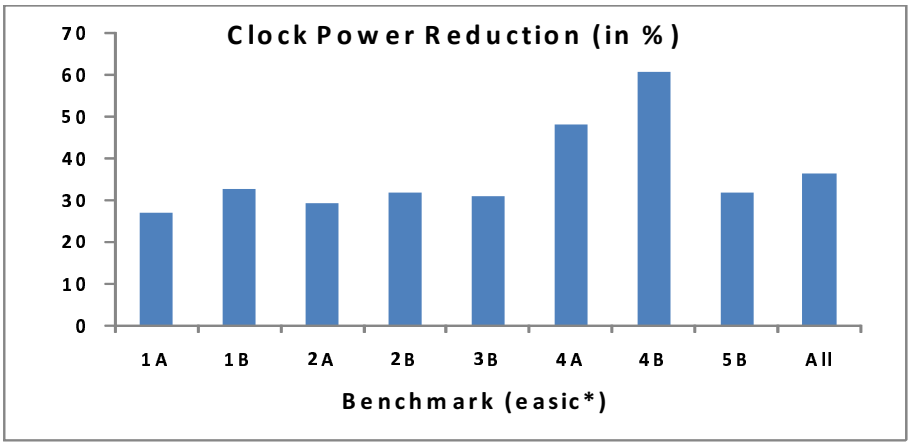

Figure 4.9: Clock power savings for placement generated by PASAP compared to RegPlace.

4.3.7 Discussions

In this section, we proposed power aware structured ASIC placement (PASAP) flow. For a given structured ASIC architecture, PASAP minimizes the leakage power and clock power dissipation by maximizing the devices that can be powered down and reducing the switched capacitance of clock network, respectively. We proposed algorithms to minimize the number of clock spines, tiles (the building block of structured ASIC platform) and columns that need to be powered on, without significantly impacting the wirelength of the design. Experimental results on large testcases show that PASAP can reduce the leakage and clock power consumption by as much as 40% and 60% respectively with 17% penalty in wirelength and 30% longer placer runtime.

Table 4.8: Increase in number of spine, tile and column that can be powered down for various benchmarks.

Bench Name	# Unused Spines			# Unused Tiles			# Unused Columns			Wirelength (x10^6)			CPU Runtime (s)		
	RP	PA	Δ%	RP	PA	Δ%	RP	PA	Δ%	RP	PA	Δ%	RP	PA	Δ%
easic1A	0	0	0	0	10	inf	12	1940	16066	13.2	15.5	17.3	8747	9968	13.9
easic1B	5	8	60.0	48	89	85.4	3112	7829	151	13.7	17.0	23.9	9984	11076	10.9
easic2A	0	3	inf	8	33	312.5	1086	3748	245	24.8	28.8	15.7	5974	8495	42.2
easic2B	8	12	50.0	88	119	35.2	5728	9163	60	26.1	29.6	13.3	8503	10348	21.7
easic3B	0	0	0	11	52	372.7	814	5285	549	19.7	22.4	13.9	5115	7098	38.7
easic4A	35	53	51.4	325	369	13.5	23185	24056	8.4	2.1	2.8	34.3	543	825	51.7
easic4B	38	56	47.3	417	455	9.1	25473	27014	2.8	2.2	3.0	38.5	562	741	31.9
easic5B	0	4	inf	19	54	184.2	2265	358	36.3	15.9	18.3	15.4	6409	10701	66.9
Total:	86	136	58.1	916	1179	28.7	59675	84837	42.2	118.0	137.8	16.7	45838	59253	29.3

Chapter 5

Conclusions

This dissertation studied different methods of mitigating the challenges in design of a high performance, reliable VLSI product. For achieving high performance, the dependence of the mobility of charge carrier on mechanical stress in the substrate was exploited at placement and synthesis stages. In Chapter 2, a cell level timing model was derived and used for late placement stage layout optimization. A linear program (LP) was formulated such that all timing critical cells compete among each other to grab whitespace around themselves such that higher stress version of the cell can be replaced. Next, the abstraction of stress aware physical design was raised by one more level and gate sizing and repeater insertion algorithms were proposed that concurrently consider the impact of mechanical stress. Our results show that significant delay reduction can be achieved for interconnect delay dominated using stress aware repeater insertion as well as for gate delay dominated circuits using stress aware gate sizing.

The severity of NBTI degradation and its impact on circuit delay as well as clock skew was demonstrated in Chapter 3. The reliability in terms of aging of the circuit was improved by modifying the clock gating implemen-

tation strategy and burn-in testing recipe. A runtime correction technique was presented that steers the duty cycle of the whole clock tree to be uniform irrespective of differing clock gating probabilities. For the case when the clock gating probabilities are known apriori, a design time technique was presented to intelligently select NAND or NOR based integrated clock gating (ICG) elements such that the difference in insertion delay (thus skew) is minimized. Finally, to mitigate the NBTI degradation during high stress conditions of burn-in testing, the problem of appropriately choosing the input vector during static burn-in was addressed.

Device delay and reliability are strongly dependent on feature dimensions as gate oxide thickness, channel length, contact position, etc. To perform high fidelity delay and reliability analysis or optimization, such variability needs to be tackled. Physical implementation using regular fabrics is one possible method, however to make it attractive the power and performance penalty of using regular fabrics needs to be reduced. In this dissertation, placement algorithms for high quality and low power physical implementation of designs on structured fabrics were proposed to close the performance and power gap between structured ASICs and custom/ASIC implementations. In Chapter 4 a placement algorithm considering strict clock locality, physical overlap and site constraints, was presented. To reduce the clock and leakage power, power aware placement algorithm was also proposed that can reduce the power dissipation of circuit implementation using structured ASIC.

We hope that this dissertation will foster further research follow-ups

in the above domains. Some of the possible directions that may be explored include:

- Currently through-silicon-via (TSV) based integration for realizing 3-Dimensional (3D) designs is being vigorously researched. Manufacturing of TSV impart significant tensile stress in its vicinity which opens several very interesting analysis and optimization issues. In particular, TSV induced stress aware clock tree design, detailed placement, and electromigration are some of the key issues that should be researched in future.

- The NBTI aging during dynamic burn-in testing as well as its dependence on burn-in recipe can be exploited to further reduce circuit aging during burn-in testing. In particular, the problem of choosing optimal elevated burn-in temperature and voltage to reduce NBTI degradation without impacting the time acceleration achieved by burn-in testing, can be explored. Similarly, new techniques can be explored to selectively prune out input vectors during dynamic burn-in testing such that delay degradation of design under test is reduced.

Bibliography

[1] http://www.dscc.dla.mil/Programs/MilSpec/.

[2] Corporate website. http://www.incal.com/.

[3] Corporate Websites of AMD, Intel and IBM.

[4] eASIC Corporate Website. http://www.easic.com/.

[5] GNU Linear Programming Kit. http://www.gnu.org/software/glpk/.

[6] GNU Linear Programming Toolkit. available online at http://www.gnu.org/software/glpk.

[7] *IWLS 1991 LGSynth Benchmarks.* http://www.cbl.ncsu.edu/benchmarks/LGSynth91.

[8] *Lithography Report, pp 17.* http://www.itrs.net/Links/2007ITRS/2007_Chapters.

[9] Michigan Wolverine Placer by Dongjin Lee and Igor Markov.

[10] *MOSEK Optimization Tool Manual.* http://www.mosek.com.

[11] Opensource 45-nm Digital Cell Library. available online at http://www.nangate.com/.

[12] Opensource Core Design Benchmarks. available online at http://www.opencores.org.

[13] SOC Central. http://www.soccentral.com/.

[14] The Esteem Placer, written by Bob Erickson, an independent software consultant. http://www.linkedin.com/in/boberickson.

[15] Process integration, devices and structure chapter. In *ITRS Report*. available online at http://www.itrs.net, 2007.

[16] M. A. Alam and S. Mahapatra. A Comprehensive Model for PMOS NBTI Degradation. *Microelectronic Reliability*, pages 71–81, Sept. 2005.

[17] M.A. Alam, H. Kufluoglu, D. Varghese, and S. Mahapatra. A Comprehensive Model for PMOS NBTI Degradation: Recent Progress. *Microelectronics Reliability*, 47(6):853 – 862, 2007.

[18] Charles J. Alpert, Dinesh P. Mehta, and Sachin S. Sapatnekar. *Handbook of Algorithms for Physical Design Automation*. Auerbach Publications, Boston, MA, USA, 2008.

[19] F. Andrieu, T. Ernst, F. Lime, F. Rochette, K. Romanjek, S. Barraud, C. Ravit, F. Boeuf, M. Jurczak, M. Casse, O. Weber, L. Brevard, G. Reimbold, G. Ghibaudo, and S. Deleonibus. Experimental and Comparative Investigation of Low and High Field Transport in Substrate- and Process-induced Strained nanoscaled MOSFETs. *Symposium on VLSI Technology*, pages 176–177, June 2005.

[20] Kah Wee Ang, King Jien Chui, V. Bliznetsov, Anyan Du, N. Balasubramanian, Ming Fu Li, Ganesh Samudra, and Yee-Chia Yeo. Enhanced

Performance in 50nm n-mosfets with Silicon-carbon Source/Drain Regions. *International Electron Devices Meeting*, pages 1069–1071, Dec. 2004.

[21] P. Babighian, L. Benini, and E. Macii. A Scalable Algorithm for RTL Insertion of Gated Clocks Based on ODCs Computation. *IEEE Transactions on Computer-Aided Design*, 24(1):29–42, Jan. 2005.

[22] Vaughn Betz and Jonathan Rose. VPR: A New Packing, Placement and Routing Tool for FPGA Research. In *Workshop on Field-Programmable Logic and Applications*, pages 213–222. Springer-Verlag, 1997.

[23] S. Bhardwaj, Wenping Wang, R. Vattikonda, Yu Cao, and S. Vrudhula. Predictive Modeling of the NBTI Effect for Reliable Design. *Custom Integrated Circuits Conference*, pages 189–192, Sept. 2006.

[24] R.A. Bianchi, G. Bouche, and O. Roux-dit Buisson. Accurate Modeling of Trench Isolation Induced Mechanical Stress Effects on MOSFET electrical performance. *International Electron Devices Meeting*, pages 117–120, 2002.

[25] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, March 2004.

[26] Cadence Inc. User manuals for encounter tool version 6.2. 2007.

[27] A. Chakraborty, K. Duraisami, A. Sathanur, P. Sithambaram, L. Benini, A. Macii, E. Macii, and M. Poncino. Dynamic thermal clock skew

compensation using tunable delay buffers. In *International Symposium on Low Power Electronics and Design*, pages 162–167, 2006.

[28] A. Chakraborty, S.X. Shi, and D.Z. Pan. Layout Level Timing Optimization by Leveraging Active Area Dependent Mobility of Strained-Silicon Devices. *Design, Automation and Test in Europe*, pages 849–855, March 2008.

[29] Ashutosh Chakraborty, Gokul Ganesan, Anand Rajaram, and David Z. Pan. Analysis and Optimization of NBTI Induced Clock Skew in Gated Clock Trees. In *Design, Automation and Test in Europe*, pages 296–299. IEEE, 2009.

[30] Ashutosh Chakraborty, Anurag Kumar, and David Z. Pan. RegPlace: A High Quality Open-source Placement Framework for Structured ASICs. In *Design Automation Conference*, pages 442–447, 2009.

[31] Ashutosh Chakraborty and David Z. Pan. Skew Management of NBTI Impacted Gated Clock Trees. In *International Symposium on Physical Design*, pages 127–133, 2010.

[32] Ashutosh Chakraborty and David Z. Pan. Skew Management of NBTI Impacted Gated Clock Trees. In *International Symposium on Physical Design*, pages 127–133, 2010.

[33] Ashutosh Chakraborty, Sean X. Shi, and David Z. Pan. Layout Level Timing Optimization by Leveraging Active Area Dependent Mobility of

Strained-Silicon Devices. In *Design, Automation and Test in Europe*, pages 849–855, March 2008.

[34] Tony F. Chan, Jason Cong, Joseph R Shinnerl, Kenton Sze, and Min Xie. mPL6: Enhanced Multilevel Mixed-size Placement. In *International Symposium on Physical design*, pages 212–214, 2006.

[35] G. Chen, K.Y. Chuah, M.F. Li, D.S.H. Chan, C.H. Ang, J.Z. Zheng, Y. Jin, and D.L. Kwong. Dynamic NBTI of PMOS Transistors and its Impact on Device Lifetime. In *International Reliability Physics Symposium*, pages 196–202, March-4 April 2003.

[36] Minsik Cho, S. Ahmed, and D.Z. Pan. Taco: temperature aware clock-tree optimization. *International Conference on Computer-Aided Design*, pages 582–587, Nov. 2005.

[37] Brian T. Cline, Vivek Joshi, Dennis Sylvester, and David Blaauw. STEEL: A Technique for Stress-enhanced Standard Cell Library Design. In *International Conference on Computer-Aided Design*, pages 691–697, Piscataway, NJ, USA, 2008. IEEE Press.

[38] John Maxwell Cohn. Method for Reducing Design Effect of Wearout Mechanisms on Signal Skew in Integrated Circuit Design, US Patent No 6651230, November 2003.

[39] F. Crupi, C. Pace, G. Cocorullo, G. Groeseneken, M. Aoulaiche, and M. Houssa. Positive Bias Temperature Instability in nMOSFETs with

Ultra-thin Hf-silicate Gate Dielectrics. *Microelectronics Engineering*, 80(1):130–133, 2005.

[40] Monica Donno, Alessandro Ivaldi, Luca Benini, and Enrico Macii. Clock-tree Power Optimization based on RTL Clock-Gating. In *Design Automation Conference*, pages 622–627, 2003.

[41] Ken Eguro, Scott Hauck, and Akshay Sharma. Architecture-adaptive Range Limit Windowing for Simulated Annealing FPGA Placement. In *Design Automation Conference*, pages 439–444. ACM, 2005.

[42] G. Eneman, P. Verheyen, R. Rooyackers, F. Nouri, L. Washington, R. Schreutelkamp, V. Moroz, L. Smith, An De Keersgieter, M. Jurczak, and Kristin De Meyer. Scalability of the $Si_{1-x}Ge_x$ Source/Drain Technology for the 45-nm Technology Node and Beyond. *IEEE Transactions on Electron Devices*, 53(7):1647–1656, July 2006.

[43] Thomas Feudel and Manfred Horstmann. Recent Advances in Stress and Activation Engineering for high-performance Logic Transistors. *Advanced Thermal Processing of Semiconductors, 2008. RTP 2008. 16th IEEE International Conference on*, pages 1–34, 30 2008-Oct. 3 2008.

[44] Eugene A. Fitzgerald, Minjoo L. Lee, Christopher W. Leitz, and Dimitri A. Antoniadis. MOSFET Channel Engineering Using Strained Si, SiGe and Ge Channel. *Advanced Materials for Micro- and Nano-Systems*, April 2003.

[45] Robert Fourer, David M. Gay, and Brian W. Kernighan. *AMPL: A Modeling Language for Mathematical Programming.* Duxbury Press, November 2002.

[46] C. Gallon. Electrical Analysis of External Mechanical Stress Effects in Short Channel MOSFETs on (001) Silicon. *Solid State Electronics*, 48:561–566, April 2004.

[47] Feng Gao and J. P. Hayes. Exact and Heuristic Approaches to Input Vector Control for Leakage Power Reduction. In *International Conference on Computer-Aided Design*, pages 527–532, Washington, DC, USA, 2004. IEEE Computer Society.

[48] M.D. Giles, M. Armstrong, C. Auth, S.M. Cea, T. Ghani, T. Hoffmann, R. Kotlyar, P. Matagne, K. Mistry, R. Nagisetty, B. Obradovic, R. Shaheed, L. Shifren, M. Stettler, S. Tyagi, X. Wang, C. Weber, and K. Zawadzki. Understanding Stress Enhanced Performance in Intel 90nm CMOS Technology. *Digest of Technical Papers VLSI Technology*, pages 118–119, June 2004.

[49] S. Goto. An Efficient Algorithm for the Two-Dimensional Placement Problem in Electrical Circuit layout. *IEEE Transactions on Circuits and Systems*, 28(1):12–18, Jan 1981.

[50] Kanupriya Gulati, Nikhil Jayakumar, Sunil P. Khatri, and D. M. H. Walker. A Probabilistic Method to Determine the Minimum Leakage

Vector for Combinational Designs in the Presence of Random PVT Variations. *Integration VLSI Journal*, 41(3):399–412, 2008.

[51] Gurobi Optimization. Solver gurobi, 2010.

[52] Shih-Hsu Huang, Chia-Ming Chang, Wen-Pin Tu, and Song-Bin Pan. Critical-PMOS-aware Clock Tree Design Methodology for Anti-Aging Zero Skew Clock Gating. *Asia and South Pacific Design Automation Conference*, pages 480 –485, Jan. 2010.

[53] Wei Huang, Mircea R. Stan, Kevin Skadron, Karthik Sankaranarayanan, Shougata Ghosh, and Sivakumar Velusam. Compact Thermal Modeling for Temperature-aware Design. In *Design Automation Conference*, pages 878–883, 2004.

[54] S. Huda, M. Mallick, and J.H. Anderson. Clock Gating Architectures for FPGA Power Reduction. pages 112 –118, 31 2009-sept. 2 2009.

[55] Ilog, Inc. Solver cplex, 2003.

[56] M. A. B. Jackson and E. S. Kuh. Performance-driven Placement of Cell Based ICs. In *26th Design Automation Conference*, pages 370–375, 1989.

[57] Vivek Joshi, Brian Cline, Dennis Sylvester, David Blaauw, and Kanak Agarwal. Leakage power reduction using stress-enhanced layouts. In *Design Automation Conference*, pages 912–917, 2008.

[58] Vivek Joshi, Brian Cline, Dennis Sylvester, David Blaauw, and Kanak Agarwal. Stress Aware Layout Optimization. In *Internal Symposium on Physical Design*, pages 168–174, 2008.

[59] A.B. Kahng, P. Sharma, and R.O. Topaloglu. Exploiting STI Stress for Performance. *International Conference on Computer-Aided Design*, pages 83–90, Nov. 2007.

[60] E. Karl, D. Blaauw, D. Sylvester, and T. Mudge. Reliability Modeling and Management in Dynamic Microprocessor-based Systems. *43rd Design Automation Conference*, pages 1057–1060, July 2006.

[61] K. Kasamsetty, M. Ketkar, and S.S. Sapatnekar. A New Class of Convex Functions for Delay Modeling and its Application to the Transistor Sizing Problem. *IEEE Transactions on Computer-Aided Design*, 19(7):779–788, Jul 2000.

[62] G.-A. Klutke, Peter C. Kiessler, and M. A. Wortman. A critical look at the bathtub curve. *IEEE Transactions on Reliability*, 52(1):125–129, 2003.

[63] Harold W. Kuhn. The Hungarian Method for the Assignment Problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.

[64] Sanjay V. Kumar, Chris H. Kim, and Sachin S. Sapatnekar. An Analytical Model for Negative Bias Temperature Instability. In *International Conference on Computer Aided design*, pages 493–496, 2006.

[65] S.V. Kumar, C.H. Kim, and S.S. Sapatnekar. NBTI-Aware Synthesis of Digital Circuits. In *Design Automation Conference*, pages 370–375, June 2007.

[66] S.V. Kumar, K.H. Kim, and S.S. Sapatnekar. Impact of NBTI on SRAM Read Stability and Design for Reliability. *7th International Symposium on Quality Electronic Design*, pages 6 pp.–, March 2006.

[67] Hao Li and Yue Zhuo. Criticality History Guided FPGA Placement Algorithm for Timing Optimization. In *18th ACM Great Lakes Symposium on VLSI*, pages 267–272, 2008.

[68] Wen-Shiang Liao, Yue-Gie Liaw, Mao-Chyuan Tang, Kun-Ming Chen, Sheng-Yi Huang, C.-Y. Peng, and Chee Wee Liu. PMOS Hole Mobility Enhancement Through SiGe Conductive Channel and Highly Compressive ILD SiN_x Stressing Layer. *IEEE Electron Device Letters*, 29(1):86–88, Jan. 2008.

[69] Andrew C. Ling, Deshanand P Singh, and Stephen D. Brown. Incremental Placement for Structured ASICs Using the Transportation Problem. *International Conference on Very Large Scale Integration*, pages 172–177, Oct. 2007.

[70] Tao Luo and David Z. Pan. DPlace2.0: A Stable and Efficient Analytical Placement Based on Diffusion. In *Asia and South Pacific Design Automation Conference*, pages 346–351, 2008.

[71] S. Mahapatra, D. Saha, D. Varghese, and P.B. Kumar. On the Generation and Recovery of Interface Traps in MOSFETs Subjected to NBTI, FN, and HCI stress. *IEEE Transactions on Electron Devices*, 53(7):1583 –1592, July 2006.

[72] S. Mayuzumi, J. Wang, S. Yamakawa, Y. Tateshita, T. Hirano, M. Nakata, S. Yamaguchi, Y. Yamamoto, Y. Miyanami, I. Oshiyama, K. Tanaka, K. Tai, K. Ogawa, K. Kugimiya, Y. Nagahama, Y. Hagimoto, R. Yamamoto, S. Kanda, K. Nagano, H. Wakabayashi, Y. Tagawa, M. Tsukamoto, H. Iwamoto, M. Saito, S. Kadomura, and N. Nagashima. Extreme high-performance n- and p-mosfets boosted by dual-metal/high-k gate damascene process using top-cut dual stress liners on (100) substrates. *International Electron Devices Meeting*, pages 293–296, Dec. 2007.

[73] K. Mistry, M. Armstrong, C. Auth, S. Cea, T. Coan, T. Ghani, T. Hoffmann, A. Murthy, J. Sandford, R. Shaheed, K. Zawadzki, K. Zhang, S. Thompson, and M. Bohr. Delaying Forever: Uniaxial Strained Silicon Transistors in a 90nm CMOS Technology. *VLSI Technology*, pages 50–51, June 2004.

[74] P. Morin. Mechanical Stress in Silicon Based Materials: Evolution Upon Annealing and Impact on Devices Performances. *International Conference on Advanced Thermal Processing of Semiconductors*, pages 93–102, Oct. 2006.

[75] H. Nii, T. Sanuki, Y. Okayama, K. Ota, T. Iwamoto, T. Fujimaki, T. Kimura, R. Watanabe, T. Komoda, A. Eiho, K. Aikawa, H. Yamaguchi, R. Morimoto, K. Ohshima, T. Yokoyama, T. Matsumoto, K. Hachimine, Y. Sogo, S. Shino, S. Kanai, T. Yamazak, S. Takahashi, H. Maeda, T. Iwata, K. Ohno, Y. Takegawa, A. Oishi, M. Togo, K. Fukasaku, Y. Takasu, H. Yamasaki, H. Inokuma, K. Matsuo, T. Sato, M. Nakazawa, T. Katagiri, K. Nakazawa, T. Shinyama, T. Tetsuka, S. Fujita, Y. Kagawa, K. Nagaoka, S. Muramatsu, S. Iwasa, S. Mimotogi, K. Yoshida, K. Sunouchi, M. Iwai, M. Saito, M. Ikeda, Y. Enomoto, H. Naruse, K. Imai, S. Yamada, N. Nagashima, T. Kuwata, and F. Matsuoka. A 45nm High Performance Bulk Logic Platform Technology (CMOS6) using Ultra High NA(1.07) Immersion Lithography with Hybrid Dual-Damascene Structure and Porous Low-k BEOL. *International Electron Devices Meeting*, pages 1–4, Dec. 2006.

[76] B. Obradovic, P. Matagne, L. Shifren, X. Wang, M. Stettler, J. He, and M.D. Giles. A Physically-based Analytic Model for Stress-induced Hole Mobility Enhancement. In *10th International Workshop on Computational Electronics*, pages 26–27, Oct 2004.

[77] Takumi Okamoto, Tsutomu Kimoto, and Naotaka Maeda. Design Methodology and Tools for NEC Electronics' Structured ASIC ISSP. In *International Symposium on Physical design*, pages 90–96, 2004.

[78] Min Pan, N. Viswanathan, and C. Chu. An Efficient and Effective De-

tailed Placement Algorithm. In *International Conference on Computer-Aided Design*, pages 48 – 55, Nov. 2005.

[79] Jan M. Rabaey, Anantha Chandrakasan, and Borivoje Nikolic. *Digital Integrated Circuits*. Prentice Hall Press, 2nd edition, 2003.

[80] Arjun Rajagopal. Clock Tree Design Challenges for Robust and Low Power Design. In *International Symposium on Physical Design*, pages 168–168, 2006.

[81] Anand Rajaram, David Z. Pan, and Jiang Hu. Improved Algorithms for link-based non-tree Clock Networks for Skew Variability Reduction. In *International Symposium on Physical Design*, pages 55–62, 2005.

[82] Jarrod A. Roy, David A. Papa, Saurabh N. Adya, Hayward H. Chan, Aaron N. Ng, James F. Lu, and Igor L. Markov. CAPO: Robust and Scalable Open-source min-cut Floorplacer. In *International Symposium on Physical design*, pages 224–226, 2005.

[83] Hernan A. Rueda. *Modeling of Mechanical Stress in Silicon Isolation Technology and Its Influence on Device Characteristics*. PhD thesis, University of Florida, 1999.

[84] Herman Schmit, Amit Gupta, and Radu Ciobanu. Placement Challenges for Structured ASICs. In *International Symposium on Physical design*, pages 84–86, 2008.

[85] Oleg Semenov, Arman Vassighi, Manoj Sachdev, Ali Keshavarzi, and C.F. Hawkins. Burn-in temperature projections for deep sub-micron technologies. *International Test Conference*, page 95, 2003.

[86] E.M. Sentovich, K.J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P.R. Stephan, Robert K. Brayton, and Alberto L. Sangiovanni-Vincentelli. Sis: A system for sequential circuit synthesis. Technical Report UCB/ERL M92/41, EECS Department, University of California, Berkeley, 1992.

[87] L. Shifren, X. Wang, P. Matagne, B. Obradovic, C. Auth, S. Cea, T. Ghani, J. He, T. Hoffman, R. Kotlyar, Z. Ma, K. Mistry, R. Nagisetty, R. Shaheed, M. Stettler, C. Weber, and M. D. Giles. Drive Current Enhancement in p-type Metal-Oxide-Semiconductor Field-Effect Transistors Under Shear Uniaxial Stress. *Applied Physics Letters*, 85(25):6188–6190, Dec 2004.

[88] Guangyu Sun, Yongke Sun, Toshikazu Nishida, and Scott E. Thompson. Hole Mobility in Silicon Inversion Layers: Stress and Surface Orientation. *Journal of Applied Physics*, 102(8):084501–084501–7, Oct 2007.

[89] Synopsys. *HSpice User Guide*. 2002.

[90] R. Reed Taylor and Herman Schmit. Creating a Power-Aware Structured ASIC. *International Symposium on Low Power Electronics and Design*, 0:74–77, 2004.

[91] S. Thompson, N. Anand, M. Armstrong, C. Auth, B. Arcot, M. Alavi, P. Bai, J. Bielefeld, R. Bigwood, J. Brandenburg, M. Buehler, S. Cea, V. Chikarmane, C. Choi, R. Frankovic, T. Ghani, G. Glass, W. Han, T. Hoffmann, M. Hussein, P. Jacob, A. Jain, C. Jan, S. Joshi, C. Kenyon, J. Klaus, S. Klopcic, J. Luce, Z. Ma, B. Mcintyre, K. Mistry, A. Murthy, P. Nguyen, H. Pearson, T. Sandford, R. Schweinfurth, R. Shaheed, S. Sivakumar, M. Taylor, B. Tufts, C. Wallace, P. Wang, C. Weber, and M. Bohr. A 90-nm Logic Technology Featuring Strained-Silicon. *IEEE Transactions on Electron Devices*, 51(11):1790–1797, Nov. 2004.

[92] Rakesh Vattikonda, Yansheng Luo, Alex Gyure, Xiaoning Qi, Sam Lo, Mahmoud Shahram, Yu Cao, Kishore Singhal, and Dino Toffolon. A New Simulation Method for NBTI Analysis in SPICE Environment. *International Symposium on Quality Electronic Design*, pages 41–46, March 2007.

[93] Rakesh Vattikonda, Wenping Wang, and Yu Cao. Modeling and Minimization of PMOS NBTI Effect for Robust Nanometer Design. In *43rd Annual Design Automation Conference*, pages 1047–1052, 2006.

[94] N. Viswanathan, Min Pan, and C. Chu. FastPlace 3.0: A Fast Multilevel Quadratic Placement Algorithm with Placement Congestion control. In *Asia South Pacific Design Automation Conference*, pages 135–140, 2007.

[95] Qifa W., Yujing T., Wei X., and Tomisawa M. Achieve Low Power Design with Power Compiler. *SNUG, OKI Techno Centre, Singapore*,

2002.

[96] Li Wang, Matthew French, Azadeh Davoodi, and Deepak Agarwal. FPGA Dynamic Power Minimization Through Placement and Routing Constraints. *EURASIP Journal of Embedded System*, 2006:7–7, Jan 2006.

[97] Qiang Wang, Subodh Gupta, and Jason H. Anderson. Clock Power Reduction for Virtex-5 FPGAs. In *ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, FPGA '09, pages 13–22, 2009.

[98] Wenping Wang, Shengqi Yang, S. Bhardwaj, R. Vattikonda, S. Vrudhula, F. Liu, and Yu Cao. The Impact of NBTI on the Performance of Combinational and Sequential Circuits. *44th ACM/IEEE Design Automation Conference*, pages 364–369, June 2007.

[99] Yu Wang, Hong Luo, Ku He, Rong Luo, Huazhong Yang, and Yuan Xie. Temperature-aware NBTI Modeling and the Impact of Input Vector Control on Performance Degradation. In *Design, Automation and Test in Europe*, pages 546–551, San Jose, CA, USA, 2007. EDA Consortium.

[100] L. Washington, F. Nouri, S. Thirupapuliyur, G. Eneman, P. Verheyen, V. Moroz, L. Smith, Xiaopeng Xu, M. Kawaguchi, T. Huang, K. Ahmed, M. Balseanu, Li-Qun Xia, Meihua Shen, Yihwan Kim, R. Rooyackers,

Kristin De Meyer, and R. Schreutelkamp. pMOSFET with 200% Mobility Enhancement Induced by Multiple Stressors. *IEEE Electron Device Letters*, 27(6):511–513, June 2006.

[101] M. Wiatr, T. Feudel, A. Wei, A. Mowry, R. Boschke, P. Javorka, A. Gehring, T. Kammler, M. Lenski, K. Frohberg, R. Richter, M. Horstmann, and D. Greenlaw. Review on Process-Induced Strain Techniques for Advanced Logic Technologies. In *15th International Conference on Advanced Thermal Processing of Semiconductors*, pages 19–29, Oct. 2007.

[102] H.S. Yang, R. Malik, S. Narasimha, Y. Li, R. Divakaruni, P. Agnello, S. Allen, A. Antreasyan, J.C. Arnold, K. Bandy, M. Belyansky, A. Bonnoit, G. Bronner, V. Chan, X. Chen, Z. Chen, D. Chidambarrao, A. Chou, W. Clark, S.W. Crowder, B. Engel, H. Harifuchi, S.F. Huang, R. Jagannathan, F.F. Jamin, Y. Kohyama, H. Kuroda, C.W. Lai, H.K. Lee, W.-H. Lee, E.H. Lim, W. Lai, A. Mallikarjunan, K. Matsumoto, A. McKnight, J. Nayak, H.Y. Ng, S. Panda, R. Rengarajan, M. Steigerwalt, S. Subbanna, K. Subramanian, J. Sudijono, G. Sudo, S.-P. Sun, B. Tessier, Y. Toyoshima, P. Tran, R. Wise, R. Wong, I.Y. Yang, C.H. Wann, L.T. Su, M. Horstmann, Th. Feudel, A. Wei, K. Frohberg, G. Burbach, M. Gerhardt, M. Lenski, R. Stephan, K. Wieczorek, M. Schaller, H. Salz, J. Hohage, H. Ruelke, J. Klais, P. Huebler, S. Luning, R. van Bentum, G. Grasshoff, C. Schwan, E. Ehrichs, S. Goad, J. Buller, S. Krishnan, D. Greenlaw, M. Raab, and N. Kepler. Dual stress liner for high perfor-

mance sub-45nm gate length SOI CMOS manufacturing. *International Electron Devices Meeting*, pages 1075–1077, Dec. 2004.

[103] Yee-Chia Yeo, Qiang Lu, Tsu-Jae King, Chenming Hu, T. Kawashima, M. Oishi, S. Mashiro, and J. Sakai. Enhanced Performance in sub-100 nm CMOSFETs using Strained Epitaxial Silicon-Germanium. *International Electron Devices Meeting*, pages 753–756, 2000.

[104] Mohd Fairuz Zakaria, Zainal Abu Kassim, Melanie Po-Leen Ooi, and Serge Demidenko. Reducing burn-in time through high-voltage stress test and weibull statistical analysis. *IEEE Design and Test of Computers*, 23(2):88–98, 2006.

[105] Bin Zhang and M. Orshansky. Modeling of NBTI-Induced PMOS Degradation under Arbitrary Dynamic Temperature Variation. *International Symposium on Quality Electronic Design*, pages 774–779, March 2008.

[106] Yue Zhuo, Hao Li, Qiang Zhou, Yici Cai, and Xianlong Hong. New Timing and Routability Driven Placement Algorithms for FPGA Synthesis. In *ACM Great Lakes Symposium on VLSI*, pages 570–575. ACM, 2007.

Vita

Ashutosh Chakraborty was born on Sept 28, 1981 in New Delhi, India. He did his schooling at various Central Government *Kendriya Vidyalas* in New Delhi, Kabul (Afghanistan) and BHEL Haridwar. He received his B.Tech. in Electrical Engineering from Indian Institute of Technology, New Delhi, India in 2002 and M.S. degree in Computer Engineering from The University of Texas at Austin in 2008. From 2002 to 2004, he worked as a Senior Member Technical Staff at Mentor Graphics (India) on hardware emulation products. From 2004 to 2006, he was a Research Assistant at Politecnico di Torino, Italy working on low power design. He is currently in the PhD program at University of Texas at Austin. He has published over 20 refereed papers in international conferences and journals. He interned at IKOS (New Delhi, India) during summer 2001, Advanced Micron Devices (Austin, TX) during summer 2007, Synopsys Inc. (Hillsboro, OR) during summer and fall 2009, and IBM TJ Watson Research Centre (Yorktown Heights, NY) during summer 2010.

Ashutosh's research interests include CAD algorithms for improving reliability, performance and yield of nanometer IC designs. In particular, he has worked on algorithms for repeater insertion, temperature aware design, clock tree synthesis, physical design for regular fabrics and stress aware placement among other projects. Ashutosh received best Interactive Presentation

award at DATE 2009, best paper nomination at ISPD(2010), eASIC placement contest Grand Prize (2009), University of Texas Fellowship (2009), Govt of Italy fellowship (2004) and Rajiv Bhambhavle best undergraduate thesis award (2002) as well as several travel grants. He has served as reviewer for several international conferences and journals including DAC, SLIP, TCAD, TVLSI, TCAS-I, TCAS-II, JSCS.

Permanent address: C/O Mr. M. K. Chakraborty,
CB 41D, M.I.G. Flats,
Opp. R.B.I. Colony,
Shalimar Bagh,
New Delhi 110088, INDIA

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.